RM0461
Reference manual

STM32WLEx advanced Arm®-based 32-bit MCUs with sub-GHz radio solution

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32WLEx microcontrollers memory and peripherals.

STM32WLEx microcontrollers with integrated sub-GHZ radio operating in the 150 - 960 MHz ISM band, belong to a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics, refer to the corresponding datasheets.

For information on the Arm® Cortex®-M4 core, refer to the corresponding Arm® Technical Reference Manuals available on http://infocenter.arm.com.

STM32WLEx microcontrollers include ST state-of-the-art patented technology.

Related documents

- STM32WLE5xx STM32WLE4xx datasheet (DS13105)

For information on the device errata with respect to the datasheet and reference manual, refer to the STM32WLE5xx STM32WLE4xx errata sheet (ES0506).

Contents

List of tables

List of figures

1 Documentation conventions

1.1 General information

The STM32WLEx devices embed an Arm®(a) Cortex®-M4 with DSP.

arm

1.2 List of abbreviations for registers

The following abbreviations(b) are used in register descriptions:

read/write (rw)	Software can read and write to this bit.
read-only (r)	Software can only read this bit.
write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
read/clear write0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value.
read/clear write1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
read/clear write (rc_w)	Software can read as well as clear this bit by writing to the register. The value written to this bit is not important.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to 0. Writing this bit has no effect on the bit value.
read/set by read (rs_r)	Software can read this bit. Reading this bit automatically sets it to 1. Writing this bit has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing 0 has no effect on the bit value.
read/write once (rwo)	Software can only write once to this bit and can also read it at any time. Only a reset can return the bit to its reset value.
toggle (t)	The software can toggle this bit by writing 1. Writing 0 has no effect.
read-only write trigger (rt_w1)	Software can read this bit. Writing 1 triggers an event but has no effect on the bit value.
Reserved (Res.)	Reserved bit, must be kept at reset value.

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

b. This is an exhaustive list of all abbreviations applicable to STMicroelectronics microcontrollers, some of them may not be used in the current document.

1.3 Glossary

This section gives a brief definition of acronyms and abbreviations used in this document:

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **Option bytes**: product configuration bits stored in the Flash memory.
- **AHB**: advanced high-performance bus.

1.4 Availability of peripherals

For availability of peripherals and their number across all sales types, refer to the particular device datasheet.

2 Memory and bus architecture

The following definition is used in this section:

- CPU = Arm Cortex-M4 with MPU and DSP

2.1 System architecture

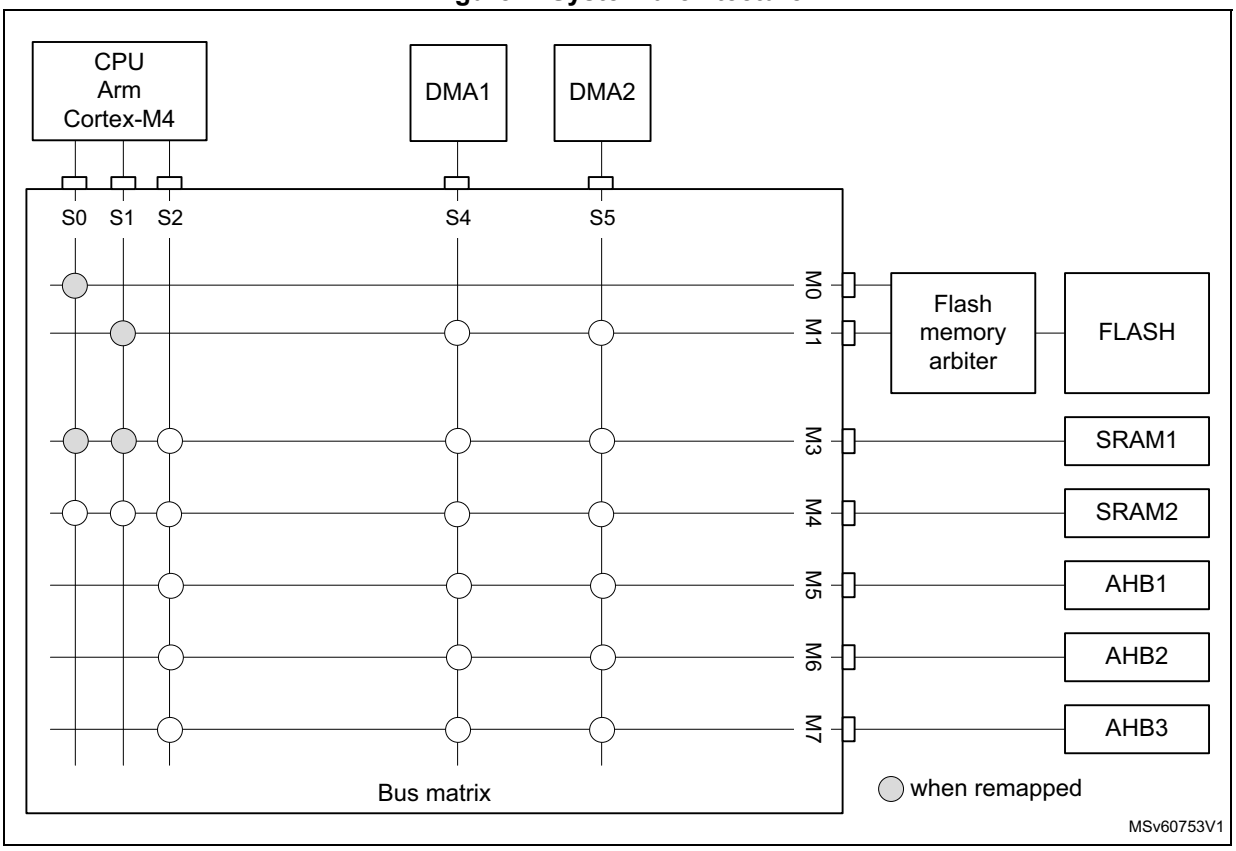
The main system consists of a 32-bit multilayer AHB bus matrix that interconnects the following masters and slaves:

- Five masters:
  - CPU core I-bus
  - CPU core D-bus
  - CPU core S-bus
  - DMA1
  - DMA2
- Eight slaves:
  - Internal flash memory on the CPU Code bus
  - Internal flash memory on CPU DCode bus
  - Internal SRAM1 (up to 32 Kbytes, size depending on the device, refer to the datasheet)
  - Internal SRAM2 (up to 32 Kbytes, size depending on the device, refer to the datasheet)
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - AHB3 peripherals including AHB to APB bridges and APB peripherals (connected to APB3)

The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously.

This architecture is shown in the figure below.

Figure 1. System architecture



2.1.1 S0: CPU I-bus

This bus connects the instruction bus of the CPU core to the bus matrix. This bus is used by the core to fetch instructions. The targets of this bus are the internal flash memory, SRAM1and SRAM2.

2.1.2 S1: CPU D-bus

This bus connects the data bus of the CPU core to the bus matrix. This bus is used by the core for literal load and debug access. The targets of this bus are the internal flash memory, SRAM1 and SRAM2.

2.1.3 S2: CPU S-bus

This bus connects the system bus of the CPU core to the bus matrix. This bus is used by the core to access data located in a peripheral or SRAM area. The targets of this bus are the SRAM1, SRAM2, the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals and the AHB3 peripherals including the APB3.

2.1.4 S4, S5: DMA-bus

These buses connect the AHB master interface of the DMAs to the bus matrix.The targets of this bus are the internal flash memory, SRAM1, SRAM2 the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals and the AHB3 peripherals including the APB3 peripherals.

AHB/APB bridges

The two bridges AHB to APB1 and AHB to ABP2 provide full synchronous connections between the AHB1 and the two APB buses, allowing flexible selection of the peripheral frequency.

The bridge AHB to APB3 provides full synchronous connections between the AHB and the APB bus, allowing flexible selection of the frequency between the AHB and peripherals.

Refer to *Section 2.4.2: Memory map and register boundary addresses* for the address mapping of the peripherals connected to this bridge.

After each device reset, all peripheral clocks are disabled, except for the SRAM1/2 and the flash memory interface. Before using a peripheral, its clock must be enabled in the RCC_AHBxENR and RCC_APBxENR registers.

Note: When a 16- or 8-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 16- or 8-bit data to feed the 32-bit vector.

2.2 Boot configuration

Three different CPU boot modes can be selected through the BOOT0 pin and nBOOT1 bit in the user options.

Boot is furthermore conditioned by the CPU boot lock enable and the user flash memory empty check, as shown in the table below.

Table 1. Device boot mode

Boot mode selection					Valid options	User Flash empty	CPU aliasing space
nBOOT1 option	nBOOT0 option	PH3/BOOT0	nSWBOOT0 option	BOOT_LOCK			
x	x	0	1	x	No	x	Hold
		1	1				SRAM1 boot
	1	x	0				Hold
	0		0				SRAM1 boot

Table 1. Device boot mode (continued)

Boot mode selection					Valid options	User Flash empty	CPU aliasing space
nBOOT1 option	nBOOT0 option	PH3/BOOT0	nSWBOOT0 option	BOOT_LOCK			
x	x	0	1	0	Yes	0	User Flash boot
						1	System Flash boot
1		1				x	System Flash boot
0							SRAM1 boot
x		x		1			User Flash boot
x	1	x	0	0		0	User Flash boot
						1	System Flash boot
1	0					x	System Flash boot
0							SRAM1 boot
x	x			1			User Flash boot

Values on BOOT0 and BOOT1 are latched after a reset. It is up to the user to provide the correct value for the required boot mode.

BOOT0 and BOOT1 are also re-sampled when exiting Standby mode. Consequently they must be kept in the required boot mode. After the startup delay, the CPU fetches the top-of-stack from address 0x0000 0000, then starts code execution from the boot memory at 0x0000 0004.

Depending on the selected boot mode, main flash memory, system flash memory and SRAM1 are accessible as follows:

- Boot from main flash memory

  The main flash memory is aliased in the CPU boot memory space at address 0x0000 0000 and is also still accessible from its physical address 0x0800 0000. In other words, the flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from system flash memory

  The system flash memory is aliased in the CPU boot memory space at address 0x0000 0000 and is also still accessible from its physical address 0x1FFF 0000.
- Boot from SRAM memory

  The SRAM memory is aliased in the CPU boot memory space at address 0x0000 0000 and is also still accessible from its physical address 0x2000 0000.

CPU SRAM physical remap

Following CPU boot, the application software can modify the memory map at address 0x0000 0000. This modification is performed by programming the SYSCFG memory remap register (SYSCFG_MEMRMP) in the SYSCFG controller.

The following memories can be remapped:

- Main flash memory
- System flash memory
- SRAM memory

Embedded bootloader

The embedded bootloader is located in the system flash memory, programmed by STMicroelectronics during production. It is used to program the flash memory using one of the following device interfaces:

- USART1 on pins PA9 and PA10
- USART2 on pins PA2 and PA3
- SPI1 on pins PA4, PA5, PA6 and PA7
- SPI2S2 on pins PB12, PB13, PB14 and PB15

The embedded bootloader runs on the CPU and can be used to load content in memory areas.

2.3 SRAM erase

SRAM1, SRAM2 and PKA SRAM provide an SRAM erase feature.

These SRAMs are erased under the conditions detailed in the table below.

Table 2. SRAM erase conditions

Condition	SRAM1(1)	SRAM2(1)	PKA SRAM(2)
System reset(3) (user option SRAM_RST = 1)	Retained	Retained	Hardware erased
System reset (user option SRAM_RST = 0)	Hardware erased	Hardware erased	Hardware erased
OBL with invalid user options	Hardware erased	Hardware erased	Hardware erased
RDP regression from 1 to 0, on OPTSTRT.	Hardware erased(4)	Hardware erased	Hardware erased
Tamper(5)	Retained	Hardware erased	Hardware erased
SYSCFG_SCSR SRAM2ER	Retained	Hardware erased	Retained

1. An ongoing SRAM1 or SRAM2 erase can be monitored by SYSCFG_SCSR.SRAMBSY flag.
2. An ongoing PKA SRAM erase can be monitored by SYSCFG_SCSR.PKASRAMBSY flag.
3. POR, NRST and wakeup from Standby.
4. For more details, see *Table 12: RDP regression from level 1 to level 0 and memory erase*.
5. To be able to debug without SRAM erase on tamper, especially ITAM6 debug access, the tamper erase must be disabled in the TAMP by firmware.

2.4 Memory organization

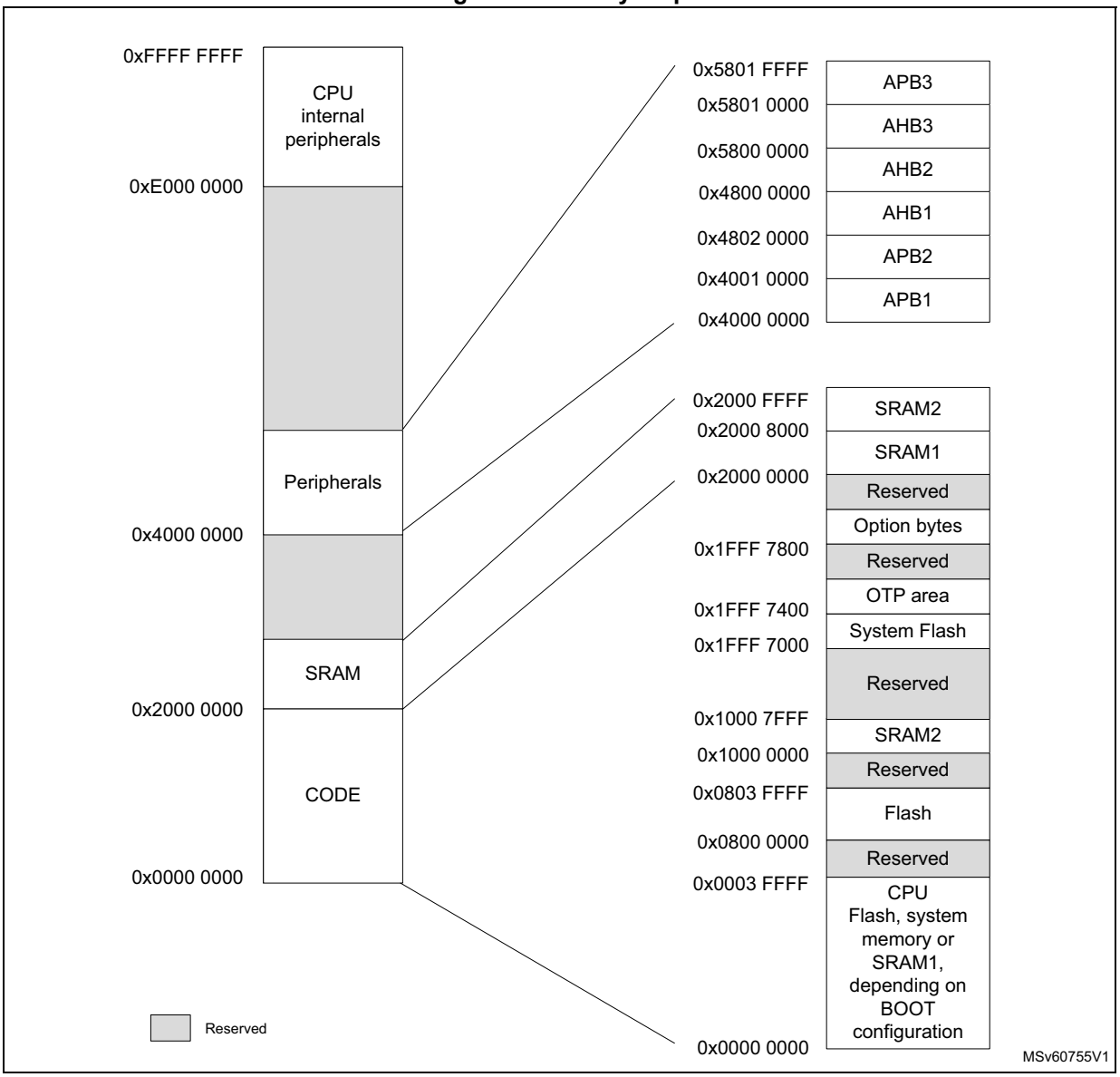
2.4.1 Introduction

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

2.4.2 Memory map and register boundary addresses

Figure 2. Memory map



Any memory area not allocated to on-chip memories and peripherals is considered “Reserved”.

The table below details the boundary addresses of peripherals available in the device.

Table 3. Memory map and peripheral register boundary addresses

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB3	0x5801 0400 - 0x5801 FFFF	-	Reserved	-
	0x5801 0000 - 0x5801 03FF	1 K	SUBGHZSPI	*Section 35.9.10: SPI/I2S register map*
AHB3	0x5800 4400 - 0x5800 FFFF	-	Reserved	-
	0x5800 4000 - 0x5800 43FF	1 K	FLASH	*Section 3.8.14: FLASH register map*
	0x5800 3400 - 0x5800 3FFF	8 K	PKA continue	*Section 22.7.5: PKA register map*
	0x5800 2400 - 0x5800 33FF		PKA RAM	
	0x5800 2000 - 0x5800 23FF		PKA	
	0x5800 1C00 - 0x5800 1FFF	-	Reserved	-
	0x5800 1800 - 0x5800 1BFF	1 K	AES	*Section 21.7.18: AES register map*
	0x5800 1400 - 0x5800 17FF	1 K	HSEM	*Section 7.4.9: HSEM register map*
	0x5800 1000 - 0x5800 13FF	1 K	True RNG	*Section 20.7.5: RNG register map*
	0x5800 0C00 - 0x5800 0FFF	-	Reserved	-
	0x5800 0800 - 0x5800 0BFF	1 K	EXTI	*Section 14.6.12: EXTI register map*
	0x5800 0400 - 0x5800 07FF	1 K	PWR	*Section 5.5.19: PWR register map*
	0x5800 0000 - 0x5800 03FF	1 K	RCC	*Section 6.4.33: RCC register map*
AHB2	0x4800 2000 - 0x57FF FFFF	-	Reserved	-
	0x4800 1C00 - 0x4800 1FFF	8 K	GPIO	*Section 8.4.36: GPIOH register map*
	0x4800 0C00 - 0x4800 1BFF			Reserved
	0x4800 0800 - 0x4800 0BFF			*Section 8.4.35: GPIOC register map*
	0x4800 0400 - 0x4800 07FF			*Section 8.4.34: GPIOB register map*
	0x4800 0000 - 0x4800 03FF			*Section 8.4.33: GPIOA register map*
AHB1	0x4260 0000 - 0x47FF FFFF	-	Reserved	-
	0x4240 0000 - 0x425F FFFF	2048 K	AHB1 bit banding	-
	0x4220 0000 - 0x423F FFFF	2048 K	APB2 bit banding	-
	0x4200 0000 - 0x421F FFFF	2048 K	APB1 bit banding	-
	0x4002 3400 - 0x41FF FFFF	-	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 K	CRC	*Section 15.4.6: CRC register map*
	0x4002 0C00 - 0x4002 2FFF	-	Reserved	-
	0x4002 0800 - 0x4002 0BFF	1 K	DMAMUX1	*Section 12.6.7: DMAMUX register map*
	0x4002 0400 - 0x4002 07FF	1 K	DMA2	*Section 11.6.7: DMA register map*
	0x4002 0000 - 0x4002 03FF	1 K	DMA1	*Section 11.6.7: DMA register map*

Table 3. Memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB2	0x4001 4C00 - 0x4001 FFFF	-	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	*Section 25.4.23: TIM16/TIM17 register map*
	0x4001 4400 - 0x4001 47FF	1 K	TIM16	*Section 25.4.23: TIM16/TIM17 register map*
	0x4001 3C00 - 0x4001 43FF	-	Reserved	-
	0x4001 3800 - 0x4001 3BFF	1 K	USART1	*Section 33.8.15: USART register map*
	0x4001 3400 - 0x4001 37FF	-	Reserved	-
	0x4001 3000 - 0x4001 33FF	1 K	SPI1	*Section 35.9.10: SPI/I2S register map*
	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	*Section 23.4.30: TIM1 register map*
	0x4001 2800 - 0x4001 2BFF	-	Reserved	-
	0x4001 2400 - 0x4001 27FF	1 K	ADC	*Section 16.13: ADC register map*
	0x4001 0400 - 0x4001 23FF	-	Reserved	-
	0x4001 0200 - 0x4001 03FF	1 K	COMP	*Section 19.6.3: COMP register map*
	0x4001 0100 - 0x4001 01FF		SYSCFG continue	*Section 9.2.12: SYSCFG register map*
	0x4001 0030 - 0x4001 00FF		VREFBUF	*Section 18.3.3: VREFBUF register map*
	0x4001 0000 - 0x4001 002F		SYSCFG	*Section 9.2.12: SYSCFG register map*

Table 3. Memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 B400 - 0x4000 FFFF	-	Reserved	-
	0x4000 B000 - 0x4000 B3FF	1 K	TAMP	*Section 31.6.11: TAMP register map*
	0x4000 9C00 - 0x4000 AFFF	-	Reserved	-
	0x4000 9800 - 0x4000 9BFF	1 K	LPTIM3	*Section 26.7.13: LPTIM register map*
	0x4000 9400 - 0x4000 97FF	1 K	LPTIM2	*Section 26.7.13: LPTIM register map*
	0x4000 8400 - 0x4000 93FF	-	Reserved	-
	0x4000 8000 - 0x4000 83FF	1 K	LPUART1	*Section 34.7.13: LPUART register map*
	0x4000 7C00 - 0x4000 7FFF	1 K	LPTIM1	*Section 26.7.13: LPTIM register map*
	0x4000 7800 - 0x4000 7BFF	-	Reserved	-
	0x4000 7400 - 0x4000 77FF	1K	DAC	*Section 17.7.16: DAC register map*
	0x4000 6000 - 0x4000 73FF	-	Reserved	-
	0x4000 5C00 - 0x4000 5FFF	1 K	I2C3	*Section 32.7.12: I2C register map*
	0x4000 5800 - 0x4000 5BFF	1 K	I2C2	*Section 32.7.12: I2C register map*
	0x4000 5400 - 0x4000 57FF	1 K	I2C1	*Section 32.7.12: I2C register map*
	0x4000 4800 - 0x4000 53FF	-	Reserved	-
	0x4000 4400 - 0x4000 47FF	1 K	USART2	*Section 33.8.15: USART register map*
	0x4000 3C00 - 0x4000 43FF	-	Reserved	-
	0x4000 3800 - 0x4000 3BFF	1 K	SPI2S2	*Section 35.9.10: SPI/I2S register map*
	0x4000 3400 - 0x4000 37FF	-	Reserved	-
	0x4000 3000 - 0x4000 33FF	1 K	IWDG	*Section 28.4.6: IWDG register map*
	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	*Section 29.5.4: WWDG register map*
	0x4000 2800 - 0x4000 2BFF	1 K	RTC	*Section 30.6.23: RTC register map*
	0x4000 0400 - 0x4000 27FF	-	Reserved	-
	0x4000 0000 - 0x4000 03FF	1 K	TIM2	*Section 24.4.25: TIMx register map*
-	0x2220 0000 - 0x3FFF FFFF	-	Reserved	-
AHB3	0x2210 0000 - 0x221F FFFF	1024 K (1)	SRAM2 bit banding	-
	0x2200 0000 - 0x220F FFFF	1024 K (1)	SRAM1 bit banding	-
-	0x2001 0000 - 0x21FF FFFF	-	Reserved	-
AHB3	0x2000 8000 - 0x2000 FFFF	32 K(1)	SRAM2	-
	0x2000 0000 - 0x2000 7FFF	32 K(1)	SRAM1	-
-	0x1FFF 8080 - 0x1FFF FFFF	-	Reserved	-

Table 3. Memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB3	0x1FFF 7800 - 0x1FFF 7FFF	2 K	Flash user options	*Section 3.8.14: FLASH register map*
	0x1FFF 7400 - 0x1FFF 77FF	1 K	Flash Engi	-
	0x1FFF 7000 - 0x1FFF 73FF	1 K	Flash OTP	-
	0x1FFF 0000 - 0x1FFF 6FFF	28 K	Flash RSS and bootloader	-
-	0x1000 8000 - 0x1FFE FFFF	-	Reserved	-
AHB3	0x1000 0000 - 0x1000 7FFF	32 K(1)	SRAM2	-
-	0x0804 0000 - 0x0FFF FFFF	-	Reserved	-
AHB3	0x0800 0000 - 0x0803 FFFF	256 K(1)	User flash	-
-	0x0004 0000 - 0x07FF FFFF	-	Reserved	-
BOOT (2)	0x0000 0000 - 0x0003 FFFF	256 K(1)	CPU boot area	-

1. Size depending on the device. Refer to the datasheet for more details.
2. Bus depends on the selected CPU boot area.

2.4.3 CPU bit banding

The CPU map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

The AHB1, APB1, APB2 peripheral registers and the SRAM1 and SRAM2 are mapped to a bit-band region, so that single bit-band write and read operations are allowed. The operations are only available for CPU accesses and not form other bus masters (such as DMA).

The peripheral bit-band alias is located from address 0x4200 0000 to 0x425F FFFF.

The SRAM bit-band alias is located from address 0x2200 0000 to 0x221F FFFF.

A mapping formula shows how to reference each word in the alias region to a corresponding bit in the bit-band region.

The mapping formula is the following:

bit_word_addr = bit_band_base + (byte_offset * 32) + (bit_number * 4)

where:

- bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- bit_band_base is the starting address of the alias region.
- byte_offset is the number of the byte in the bit_band region that contains the targeted bit.
- bit_number is the bit position (0-7) of the targeted bit.

Example

The following example shows how to map bit [2] of the byte located at SRAM1 address 0x2000 0300 to the alias region. The formula is then:

0x2200 6008 = 0x2200 0000 + 0x0300 * 32 + 2 * 4

Writing to address 0x2200 6008 has the same effect as a read-modify-write operation on bit [2] of the byte at SRAM1 address 0x2000 0300.

Reading address 0x2200 6008 returns the value 0x01 or 0x00 of bit [2] of the byte at SRAM1 address 0x2000 0300.

For more information on bit-band, refer to the Cortex-M4 programming manual.

3 Embedded flash memory (FLASH)

3.1 FLASH introduction

The flash memory interface manages the CPU AHB ICode and DCode accesses to the flash memory. It implements the access, the erase and program flash memory operations, and the read and write protection.

The flash memory interface accelerates code execution with a system of instruction prefetch and cache lines.

3.2 FLASH main features

- Up to 256 Kbytes of flash memory single bank architecture
- Memory organization: 1 bank
  - main memory: up to 256 Kbytes
  - page size: 2 Kbytes
- 72-bit wide data read (64 bits plus 8 ECC bits)
- 72-bit wide data write (64 bits plus 8 ECC bits)
- Page erase (2 Kbytes) and mass erase

Flash memory interface features:

- Flash memory read operations
- Flash memory program/erase operations
- Readout protection activated by option (RDP)
- 2 write protection areas selected by option (WRP)
- 2 proprietary code readout protection area selected by option (PCROP)
- Flash empty check
- Program and erase suspension feature
- Prefetch on CPU ICODE
- CPU instruction cache: 32 cache lines of 4 x 64 bits on ICode (1-Kbyte RAM)
- CPU data cache: 8 cache lines of 4 x 64 bits on DCode (256-byte RAM)
- Error code correction (ECC): 8 bits for 64-bit
- Option byte loader

3.3 FLASH functional description

3.3.1 Flash memory organization

The flash memory is organized as 72-bit wide memory cells (64 bits plus 8 ECC bits) that can be used for storing both code and data constants.

The flash memory is organized as follows:

- A main memory block containing 128 pages of 2 Kbytes, each page with eight rows of 256 bytes.
- An information block containing:
  - System memory from which the CPU boots in system memory boot mode

    This area is reserved and contains the bootloader used to reprogram the flash memory through one of the following interfaces: USART1, USART2, I2C1, I2C2, I2C3, SPI1, SPI2S2. It is programmed by STMicroelectronics when the device is manufactured and protected against spurious write/erase operations. For further details, refer to the application note *STM32 microcontroller system memory boot mode* (AN2606).
  - 1-Kbyte (128 double-word) OTP (one-time programmable) for user data

    The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double-word (64 bits) cannot be written anymore, even with the value 0x0000 0000 0000 0000.
  - Option bytes for user configuration

The memory organization is based on a main area and an information block as shown in the table below.

Table 4. Flash memory - Single bank organization

Area	Addresses	Size (Kbytes)	Name
Main memory	0x0800 0000 - 0x0800 07FF	2	Page 0
	0x0800 8000 - 0x0800 0FFF	2	Page 1
	0x0800 1000 - 0x0800 17FF	2	Page 2
	0x0800 1800 - 0x0800 1FFF	2	Page 3
	...	...	...
	0x083 F000 - 0x0803 F7FF	2	Page 126
	0x083 F800 - 0x0803 FFFF	2	Page 127
Information block	0x1FFF 0000 - 0x1FFF 6FFF	28	System memory
	0x1FFF 7000 - 0x1FFF 73FF	1	OTP area
	0x1FFF 7800 - 0x1FFF 7FFF	2	Option bytes

3.3.2 Empty check

During the OBL phase, after loading all options, the flash memory interface checks whether the first location of the main memory is programmed. The result of this check, in conjunction with the BOOT0 and BOOT1 information, is used to determine where the system has to boot from. It prevents the system to boot from the flash main memory area when, for example, no user code is programmed.

The flash main memory empty check status can be read from the EMPTY bit in the FLASH_ACR register. Software can modify the flash main memory empty status by writing to the EMPTY bit.

The internal empty check flag (EMPTY bit in FLASH_ACR) is implemented to allow easy programming of virgin devices by the bootloader. This flag is used when the BOOT0 pin

defines the main flash memory as target boot area. When this flag is set, the device is considered as empty and the system memory (bootloader) is selected instead of the main flash memory as a boot area, to allow the user to program the flash memory. Therefore, some of the GPIOs are reconfigured from the High-Z state. Refer to the application note AN2606 for more details concerning the bootloader and GPIO configuration in system memory boot mode. This feature can be disabled by configuring the option bytes to force a boot from the main flash memory (nSWBOOT0 = 0, nBOOT0 = 1).

This empty check flag is updated only during the loading of option bytes: it is set when the content of the address 0x08000 0000 is read as 0xFFFF FFFF, otherwise it is cleared. A power reset or setting of OBL_LAUNCH bit in FLASH_CR is needed to clear this flag after programming of a virgin device to execute user code after a system reset. The EMPTY bit can also directly be written by software.

3.3.3 Error code correction (ECC)

Data in flash memory words are 72-bit wide: eight bits are added per each double-word (64 bits).

The ECC mechanism supports the following modes:

- one error detection and correction
- two errors detection

When one error is detected and corrected, the flag ECCC (ECC correction) is set in FLASH_ECCR. If ECCCIE is set, an interrupt is generated.

When two errors are detected, the flag ECCD (ECC detection) is set in FLASH_ECCR. In this case, an NMI is generated.

When an ECC error is detected, the address of the failing double-word is saved in ADDR_ECC[16:0] in FLASH_ECCR. ADDR_ECC[2:0] bits are always cleared. The bus-ID of the CPU accessing the address is saved in CPUID[2:0].

While ECCC or ECCD is set, FLASH_ECCR is not updated if a new ECC error occurs. FLASH_ECCR is updated only when ECC flags are cleared.

Note: For a virgin data (0xFF FFFF FFFF FFFF FFFF), one error is detected and corrected, but the two errors detection mode is not supported.

When an ECC error is reported, a new read at the failing address may not generate an ECC error if the data is still present in the current buffer, even if ECCC and ECCD are cleared. If this is not the desired behavior, the user must reset the cache.

3.3.4 Read access latency

To correctly read data from the flash memory, the number of wait states (LATENCY[2:0]) must be correctly programmed in FLASH_ACR according to the frequency of the flash memory clock (HCLK3) and the internal voltage range of the device (V_{CORE}). Refer to *Section 5.1.4: Dynamic voltage scaling management*.

The table below shows the correspondence between wait states and frequency of the flash memory clock.

Table 5. Number of wait states according to flash clock (HCLK3) frequency

Wait states (WS) (access)	HCLK3 (MHz)	
	V_{CORE} range 1	V_{CORE} range 2
0 WS (1 HCLK cycle)	≤ 18	≤ 6
1 WS (2 HCLK cycles)	≤ 36	≤ 12
2 WS (3 HCLK cycles)	≤ 48	≤ 16

After power-on reset and wakeup from Standby, the HCLK3 clock frequency is 4 MHz in range 1 and 0 wait state (WS) is configured in FLASH_ACR.

When changing the frequency of the flash memory clock or the V_{CORE} range, the software sequences detailed below must be applied in order to tune the number of wait states needed to access the flash memory:

Increase the CPU frequency

1. Program the new number of wait states to the LATENCY[2:0] bits in FLASH_ACR.
2. Check that the new number of wait states is taken into account to access the flash memory by reading back the LATENCY[2:0] bits in FLASH_ACR, and wait until the new programmed number is read.
3. Modify the system cock source by writing the SW[1:0] bits in RCC_CFGR.
4. If needed, modify the CPU clock prescaler by writing the SHDHPRE[3:0] bits in RCC_EXTCFGR.
5. Optionally, check that the new system clock source or/and the new flash memory clock prescaler value is/are taken into account by reading the clock source status (SWS(1:0] bits) in RCC_CFGR, or/and the AHB prescaler value (SHDHPREF bit) in RCC_EXTCFGR.

Decrease the CPU frequency

1. Modify the system clock source by writing the SW[1:0] bits in RCC_CFGR.
2. If needed, modify the flash memory clock prescaler by writing the SHDHPRE[3:0] bits in RCC_EXTCFGR.
3. Check that the new system clock source or/and the new flash memory clock prescaler value is/are taken into account by reading the clock source status (SWS[1:0] bits) in RCC_CFGR, or/and the AHB prescaler value (SHDHPREF bit) in RCC_EXTCFGR. Wait until the new programmed system clock source or/and new flash memory clock prescaler value is/are read.
4. Program the new number of wait states to the LATENCY[2:0] bits in FLASH_ACR.
5. Optionally, check that the new number of wait states is used to access the flash memory by reading back the LATENCY[2:0] bits in FLASH_ACR.

3.3.5 Adaptive real-time memory accelerator (ART Accelerator)

The proprietary adaptive real-time (ART) memory accelerator is optimized for STM32 industry-standard Arm Cortex-M4 with DSP processors. It balances the inherent

performance advantage of the Cortex-M4 with DSP over flash memory technologies, which normally require the processor to wait for the flash memory at higher operating frequencies.

To release the processor full performance, the accelerator implements an instruction prefetch queue and branch cache that increases program execution speed from the 64-bit flash memory. Based on CoreMark® benchmark, the performance achieved thanks to the ART Accelerator is equivalent to 0 wait state program execution from the flash memory at a CPU frequency up to 48 MHz.

Instruction prefetch

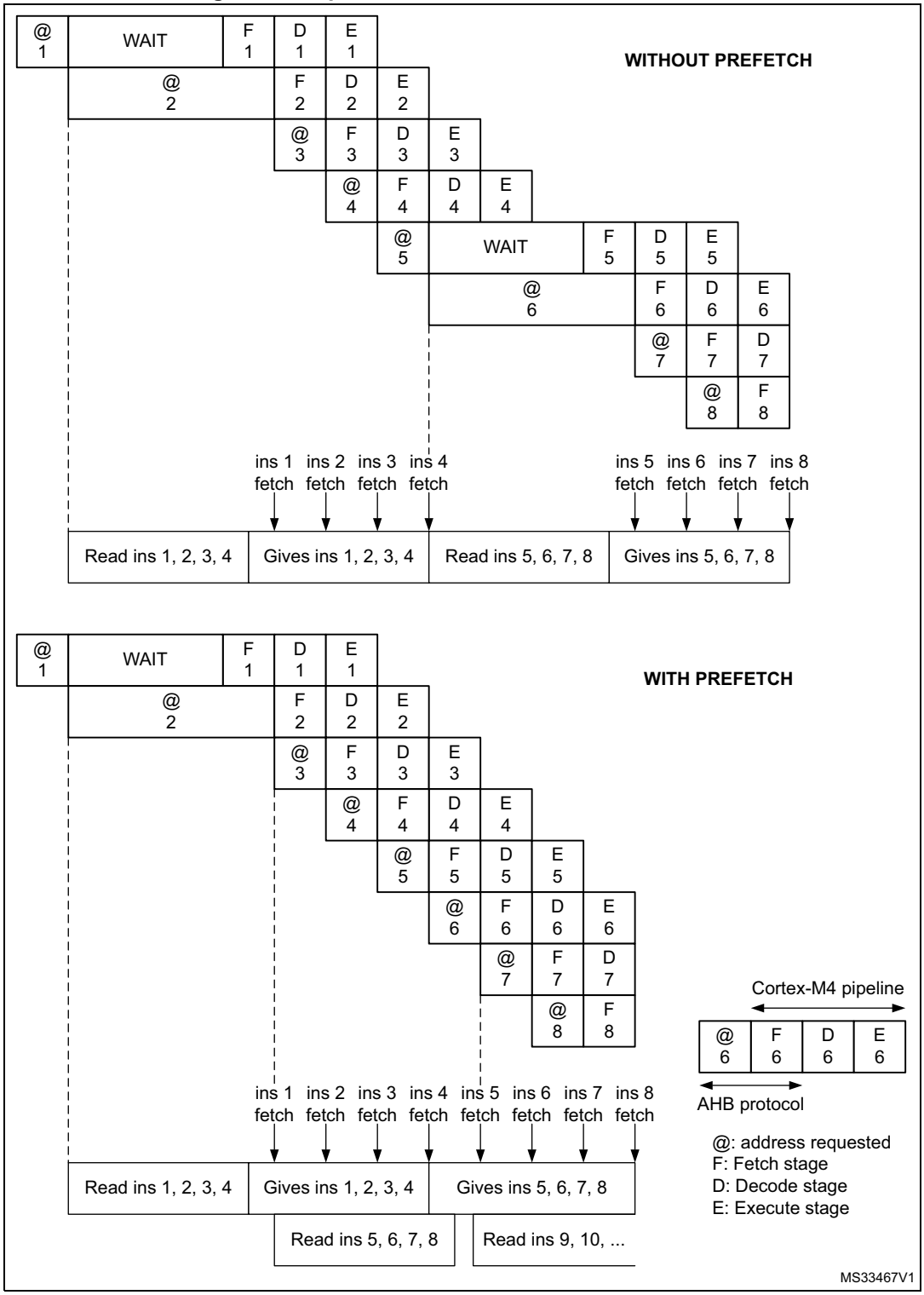
The CPU fetches the instruction over the ICode bus and the literal pool (constant/data) over the DCode bus. The prefetch block aims at increasing the efficiency of ICode bus accesses.

Each flash memory read operation provides 64 bits from either two instructions of 32 bits or four instructions of 16 bits according to the program launched. This 64-bit current instruction line is saved in a current buffer. In case of sequential code, at least two CPU cycles are needed to execute the previous read instruction line. Prefetch on the CPU ICode bus can be used to read the next sequential instruction line from the flash memory while the current instruction line is being requested by the CPU.

Prefetch is enabled by setting PRFTEN in FLASH_ACR. This feature is useful if at least one wait state is needed to access the flash memory.

The figure below shows the execution of sequential 16-bit instructions with and without prefetch when three wait states are needed to access the flash memory.

Figure 3. Sequential 16 bits instructions execution



When the code is not sequential (branch), the instruction may not be present in the currently used instruction line or in the prefetched instruction line. In this case (miss), the penalty in terms of number of cycles is at least equal to the number of wait states.

If a loop is present in the current buffer, no new access is performed.

CPU instruction cache memory (I-Cache)

To limit the CPU time lost due to jumps, it is possible to retain 32 lines of 4 x 64 bits (1 Kbyte) in an instruction cache memory.This feature is enabled for the CPU by setting the instruction cache enable (ICEN) bit in FLASH_ACR. Each time a miss occurs (requested data not present in the currently used instruction line, in the prefetched instruction line or in the instruction cache memory), the line read is copied into the instruction cache memory. If some data contained in the instruction cache memory are requested by the CPU, they are provided without inserting any delay. Once all the instruction cache memory lines have been filled, the LRU (least recently used) policy is used to determine the line to replace in the instruction memory cache. This feature is particularly useful in case of code containing loops.

The instruction cache memory is enabled after system reset.

CPU data cache memory (D-Cache)

CPU literal pools are fetched from the flash memory through the DCode bus during the execution stage of the CPU pipeline. Each CPU DCode bus read access fetches 64 bits that are saved in a current buffer. The CPU pipeline is consequently stalled until the requested literal pool is provided. To limit the time lost due to literal pools, accesses through the AHB data bus DCode have priority over accesses through the AHB instruction bus ICode.

If some literal pools are frequently used, the CPU data cache memory can be enabled by setting the data cache enable (DCEN) bit in FLASH_ACR. This feature works like the instruction cache memory but the retained data size is limited to eight lines of 4 x 64 bits (256 bytes).

The data cache memory is enabled after system reset.

Note: The D-Cache is active only when data is requested by the CPU (not by DMAs).
Data in option bytes block are not cacheable.

3.3.6 Flash program and erase operations

The embedded flash memory can be programmed using in-circuit programming or in-application programming.

The in-circuit programming (ICP) method is used to update the entire contents of the flash memory using the JTAG, SWD protocol or the supported interfaces by the system bootloader, to load the user application for the CPU, into the microcontroller. ICP offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices.

In contrast to the ICP method, in-application programming (IAP) can use any communication interface supported by the microcontroller (such as I/Os, UART, I^2C or SPI) to download programming data into memory. IAP allows the user to re-program the flash memory while the application is running. Nevertheless, part of the application must have been previously programmed in the flash memory using ICP.

The contents of the flash memory are not guaranteed if a device reset occurs during a flash memory operation.

During a program/erase operation to the flash memory, any attempt to read the flash memory stalls the bus. The read operation proceeds correctly once the program/erase operation is completed.

Unlocking the flash memory

After a reset, write is not allowed in FLASH_CR to protect the flash memory against possible unwanted operations (for example, electric disturbances).

The following sequence is used to unlock these registers:

1. Write KEY1 = 0x4567 0123 in FLASH_KEYR.
2. Write KEY2 = 0xCDEF 89AB in FLASH_KEYR.

Any wrong sequence locks up FLASH_CR until the next system reset. In the case of a wrong key sequence, a bus error is detected and a hard fault interrupt is generated.

FLASH_CR can be locked again by software by setting the LOCK bit.

Note: FLASH_CR cannot be written when BSY is set in *FLASH_SR. Any attempt to write to this register with BSY set causes the AHB bus to stall until BSY is cleared.*

3.3.7 Flash main memory erase sequences

The flash memory erase operation can be performed at page level (page erase) or on the whole memory (mass erase). Mass erase does not affect the information block (system flash memory, OTP and option bytes).

Flash memory page erase

A page erase only starts when allowed by PESD in FLASH_SR.

When a page is protected by PCROP or WRP, it is not erased.

Table 6. Page erase overview

PCROP	WRP	PCROP_RDP	Comment	WRPERR	CPU bus error
No	No	x	Page is erased	No	No
No	Yes		Page erase aborted (no page erase started)	Yes	
Yes	No				
Yes	Yes				

To erase a 2-Kbyte page, follow the steps detailed below:

1. Check that no flash memory operation is ongoing by checking BSY in FLASH_SR.
2. Check that flash program and erase operation is allowed by checking PESD in FLASH_SR.
3. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
4. Set PER and select the page to erase (PNB[6:0]) in FLASH_CR.
5. Set STRT in FLASH_CR.
6. Wait for BSY to be cleared in FLASH_SR.

Note: The internal oscillator HSI16 (16 MHz) is enabled automatically when the STRT bit is set, and disabled automatically when the STRT bit is cleared, except if the HSI16 is previously enabled with HSION in the RCC_CR register.

Flash mass erase

When PCROP or WRP is enabled, any flash memory mass erase is aborted and no erase started.

Table 7. Mass erase overview

PCROP	WRP	PCROP_RDP	Comment	WRPERR	CPU bus error
No	No	x	Memory is erased.	No	No
No	Yes		Erase aborted (no erase started)	Yes	
Yes	No				
Yes	Yes				

To perform a mass erase, follow the steps detailed below:

1. Check that no flash memory operation is ongoing by checking BSY in FLASH_SR.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set MER in FLASH_CR.
4. Set STRT in FLASH_CR
5. Wait for BSY to be cleared in FLASH_SR.

Note: The internal oscillator HSI16 (16 MHz) is enabled automatically when the STRT bit is set, and disabled automatically when the STRT bit is cleared, except if the HSI16 is previously enabled with HSION in the RCC_CR register.

3.3.8 Flash main memory programming sequences

The flash memory is programmed with 72 bits at a time (a double-word of 64 bits plus 8 bits ECC).

Programming in a previously programmed double-word is only allowed when programming an all 0 value. It is not allowed to program any other value in a previously programmed double-word. Any attempt sets PROGERR flag in FLASH_SR, except when programming an already programmed double-word with an all 0 value.

It is only possible to program a double-word (2 x 32-bit data), otherwise:

- Any attempt to write a byte (8 bits) or half-word (16 bits) sets SIZERR in FLASH_SR.
- Any attempt to write a double-word that is not aligned with a double-word address sets the PGAERR flag in FLASH_SR.

Standard programming

The flash memory programming sequence in standard mode is as follows:

1. Check that no flash main memory operation is ongoing by checking BSY in FLASH_SR.
2. Check that flash program and erase operation is allowed by checking PESD in FLASH_SR.
3. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
4. Set PG in FLASH_CR.
5. Perform the data write operation at the desired memory address, inside the main memory block or OTP area. Only double-word (64 bits) can be programmed.
   a) Write a first word in an address aligned with double-word
   b) Write the second word (see the note below)

Note: When the flash memory interface received a good sequence (a double-word), programming is automatically launched and the BSY bit is set. The internal oscillator HSI16 (16 MHz) is enabled automatically when the PG bit is set, and disabled automatically when the PG bit is cleared, except if the HSI16 is previously enabled with HSION in the RCC_CR register.
If the user needs to program only one word, double-word must be completed with the erase value 0xFFFF FFFF to launch automatically the programming.
ECC is calculated from the double-word to program.
For correct operation, the firmware must guarantee that the flash page access protection is not changed during the programming sequence. This is between the first and second word write.

6. Wait until BSY is cleared in FLASH_SR.
7. Check that EOP is set in FLASH_SR (meaning the programming operation succeeded), and clear it by software.
8. Clear PG in FLASH_SR if there no more programming request.

Fast programming

This mode allows a row to be programmed, 32 double-words (256 bytes), and the page programming time to be reduced by eliminating the need for verifying the flash memory locations before they are programmed and to avoid rising and falling time of high voltage for each double-word. During fast programming, the flash clock frequency (HCLK3) must be at least 8 MHz.

Fast row programming must be performed by executing firmware from SRAM and disabling interrupts when not relocating the CPU interrupt vector table. A read access form the CPU requesting row programming causes a bus error. A read from any other source (such as the DMA) is stalled until the row programming is finished (standard double-word programming does not cause a bus error to the requesting CPU but stalls any read until standard programming is finished).

Only the main memory can be programmed in Fast programming mode.

The flash main memory programming sequence in Fast programming mode is described below:

1. Perform a mass erase. If not, PGSERR is set.
2. Check that no flash main memory operation is ongoing by checking BSY bit FLASH_SR.
3. Check that flash memory program and erase operation is allowed by checking PESD in FLASH_SR.
4. Check and clear all error programming flag due to a previous programming.
5. Set FSTPG in FLASH_CR.
6. Write the 32 double-words to program a row (256 bytes).
7. Wait until BSY is cleared in FLASH_SR.
8. Check that the EOP flag is set in FLASH_SR(meaning the programming operation succeeded), and clear it by software.
9. Clear FSTPG in FLASH_SR if there are no more programming requests.

Note: When attempting to write in Fast programming mode while a read operation is ongoing, the programming is aborted without any system notification (no error flag is set).

When the flash memory interface receives the first double-word, programming is automatically launched. The BSY bit is set when the high voltage is applied for the first double-word, and it is cleared when the last double-word is programmed or in case of error. The internal oscillator HSI16 (16 MHz) is enabled automatically when FSTPG is set, and disabled automatically when the FSTPG bit is cleared, except if the HSI16 is previously enabled with HSION in the RCC_CR register.

The 32 double-word must be written successively. The high voltage is kept on the flash memory for all the programming. Maximum time between two double-words write requests is the time programming (around 20 µs). If a second double-word arrives after this time programming, fast programming is interrupted and MISSERR is set.

High voltage must not exceed 8 ms for a full row between two erases. This is guaranteed by the sequence of 32 double-words successively written with a clock system greater or equal to 8 MHz. An internal time-out counter counts 7 ms when Fast programming is set and stops the programming when time-out is over. In this case FASTERR is set.

If an error occurs, high voltage is stopped and next double-word to programmed is not programmed. Anyway, all previous double-words have been properly programmed.

Programming errors signaled by flags

Several kind of errors can be detected. In case of error, the flash operation (programming or erasing) is aborted.

- **PROGERR**: programming error

  In standard programming, PROGERR is set if the word to write, with a different value than all zero, is not previously erased (except if the value to program is all zero).

  If any other error occurs (such as SIZERR, PAGERR, PGSERR, WRPERR), PROGRERR may not be set even if there is a word re-programming without erase error.
- **SIZERR**: size programming error

In standard programming or in fast programming: only double-word can be programmed and only 32-bit data can be written. SIZERR is set if a byte or an half-word is written.

- **PGAERR**: alignment programming error

  PGAERR is set if one of the following conditions occurs:

  – In standard programming, the first word to be programmed is not aligned with a double-word address, or the second word does not belong to the same double-word address.

  – In fast programming, the data to program does not belong to the same row than the previous programmed double-words, or the address to program is not greater than the previous one.

- **PGSERR**: programming sequence error

  PGSERR is set if one of the following conditions occurs:

  – In the standard programming sequence or the fast programming sequence, a data is written when PG and FSTPG are cleared.

  – In the standard programming sequence or the fast programming sequence, MER and PER are not cleared when PG or FSTPG is set.

  – In the fast programming sequence, the mass erase is not performed before setting the FSTPG bit.

  – In the mass erase sequence, PG, FSTPG, and PER are not cleared when MER is set, when the security of the flash allows a mass erase. A non-secure mass erase request on a secure flash memory does not set a PGSERR. An illegal access event is generated instead.

  – In the page erase sequence, PG, FSTPG and MER are not cleared when PER is set, when the security of the page allows access. When the security of this page does not allow access, no PGSERR is set but an illegal access event is generated instead.

  – PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR or PGSERR is set due to a previous programming error.

  – In the fast programming sequence, if the page containing the row to be programmed has not been erased by the last page erase action (If a mass erase has been done, it is allowed to program rows on higher order pages but not on lower order)

- **WRPERR**: write protection error

  WRPERR is set if one of the following conditions occurs:

  – Attempt to program or erase in a write protected area (WRP) or in a PCROP area, when the security of the area allows access. When the security of this area does not allow access, no WRPERR is set but an illegal access event is generated instead.

  – Attempt to perform a mass erase when one page or more is protected by WRP or PCROP, when the security of the flash allows a mass erase. A non-secure mass erase request on a secure flash memory does not set a WRPERR. An illegal access event is generated instead.

  – The debug features are connected or the boot is executed from SRAM or from system flash memory when the readout protection (RDP) is set to level 1.

  – Attempt to modify the option bytes when the readout protection (RDP) is set to Level 2

- **MISSERR**: fast programming data miss error

  In fast programming, all the data must be written successively. MISSERR is set if the previous data programming is finished and the next data to program is not written yet.
- **FASTERR**: fast programming error

  In fast programming, FASTERR is set if one of the following conditions occurs:
  - When FSTPG bit is set for more than 7 µs (generating a time-out detection).
  - When the row fast programming is interrupted by a MISSERR, PGAERR, WRPERR or SIZERR.

If an error occurs during a program or erase operation, one of the following error flags is set in FLASH_SR:

- PROGERR, SIZERR, PGAERR, PGSERR, MISSERR (program error flags)
- WRPERR (protection error flag)

In this case, if the error interrupt enable bit ERRIE is set in FLASH_CR, an interrupt is generated and the operation error flag OPERR is set in FLASH_SR.

Note: If several successive errors are detected (for example, in case of DMA transfer to the flash memory), the error flags cannot be cleared until the end of the successive write request.

PGSERR and PGAERR in a page-based row programming

In case of fast programming, the table below describes how PGAERR and PGSERR are handled.

Table 8. Errors in page-based row programming

Last page/row	Current page/row	MER active	PPER active
page [x]/row [y]	page [x] / row [x-n]	PGAERR	PGAERR
	page [x-n] / row [any]	PGAERR & PGSERR	PGAERR & PGSERR
	page [x+n] / row [any]	No error	PGSERR

After a system reset, no MER or PER is performed. Any programming attempt causes a PGAERR and a PGSERR.

Programming errors causing a bus error

The error conditions listed below do not generate an error flag but a bus error instead:

- AHB write to any page when RDP level 1 and boot is performed from the system flash memory or SRAM1
- AHB write when the flash memory is powered down
- Read or write from the flash memory through the debugger
- Read from the flash memory when fast row programming is ongoing, for the source which requested the fast row programming
- New programming request when the previous one is not finished
- FLASH_CR register write between the two accesses of a double-word programming
- FLASH_CR register write when PESD is active (set)
- Write a wrong key in FLASH_KEYR or FLASH_OPTKEYR register
- Any subsequent write to FLASH_KEYR or FLASH_OPTKEYR after unlocking the respective feature

Programming and caches

If a flash memory write access impacts data in the data cache, the flash memory write access modifies the data in the memory and in the cache.

If an erase operation in the flash memory also concerns data in the data cache or instruction cache, the user must ensure that these data are rewritten before they are accessed during code execution. Upon an erase operation, the cache content is invalidated.

Note: The ICache and DCache must be flushed only when disabled (ICEN or DCEN = 0).

3.4 FLASH option bytes

3.4.1 Option bytes description

The option bytes can be read from the memory locations listed in the table below or from the following option byte registers:

- *FLASH option register (FLASH_OPTR)*
- *FLASH PCROP zone A start address register (FLASH_PCROP1ASR)*
- *FLASH PCROP zone A end address register (FLASH_PCROP1AER)*
- *FLASH PCROP zone B start address register (FLASH_PCROP1BSR)*
- *FLASH PCROP zone B end address register (FLASH_PCROP1BER)*
- *FLASH WRP area A address register (FLASH_WRP1AR)*
- *FLASH WRP area B address register (FLASH_WRP1BR)*

Table 9. Option bytes organization

Address[1]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1FFF 7800	Res.	BOOT_LOCK	Res.	Res.	nBOOT0	nSWBOOT0	SRAM_RST	SRAM2PAR	nBOOT1	Res.	Res.	Res.	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW	Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	BOR_LEV[2:0]			ESE	RDP[7:0]							
0x1FFF 7808	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1A_STRT[7:0]							
0x1FFF 7810	PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1A_END[7:0]							
0x1FFF 7818	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_STRT[6:0]						
0x1FFF 7820	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_STRT[6:0]						
0x1FFF 7828	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1B_STRT[7:0]							
0x1FFF 7830	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1B_END[7:0]							
0x1FFF 7838 to 0x1FFF 7FF0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x1FFF 7FF8	OPTVAL[31:0]																															

1. The upper 32 bits of the double-word address contain the inverted data from the lower 32 bits.

3.4.2 Option bytes programming

After a reset, the options related bits in FLASH_CR are write-protected. To run any operation on the option bytes page, the option lock bit OPTLOCK in FLASH_CR must be cleared.

The following sequence is used to unlock the FLASH_CR and flash memory option registers:

1. Unlock FLASH_CR with the LOCK clearing sequence (refer to *Unlocking the flash memory*).
2. Write OPTKEY1 = 0x0819 2A3B in FLASH_OPTKEYR
3. Write OPTKEY2 = 0x4C5D 6E7F in FLASH_OPTKEYR

Any wrong sequence locks up the flash option register until the next system reset. In the case of a wrong key sequence, a bus error is detected and a hard fault interrupt is generated.

The user options can be protected against unwanted erase/program operations by setting the OPTLOCK bit by software.

Note: If LOCK is set by software, OPTLOCK is automatically set as well.

Modify user options

The option bytes are programmed differently from a main memory user address.

To modify the user options value, follow the procedure below:

1. Clear OPTLOCK option lock bit with the clearing sequence described above
2. Write the desired options value in the options registers.
3. Check that no flash memory operation is ongoing by checking the BSY bit in FLASH_SR.
4. Check that flash program and erase operation is allowed by checking the PESD bit in FLASH_SR.
5. Set the options start bit OPTSTRT in FLASH_CR.
6. Wait for the BSY bit to be cleared.

Note: Any modification of the value of one option is automatically performed by erasing user option bytes pages first, and then programming all the option bytes with the values contained in the flash option registers.

Warning: On RDP regression from level 1 to level 0, starting the option programming by the OPTSTRT bit causes the flash memory, SRAM1 and SRAM2 to be erased. All device software in all memories is erased. A subsequent OBL_LAUNCH must be started by an external tool or a POR must be performed to restart the device and reload the options.

Option byte loading

After the BSY bit is cleared, all new options are updated into the flash memory but not applied to the system. A read from the option registers still returns the last loaded option byte values. The new options have effect on the system only after they are loaded.

Option bytes loading is performed in the following two cases:

- when OBL_LAUNCH bit is set in FLASH_CR
- after a power reset (BOR reset or exit from Standby/Shutdown modes)

Option byte loader performs a read of the options block and stores the data into internal option registers. These internal registers configure the system and can be read by software. Setting OBL_LAUNCH generates a reset so the option byte loading is performed under system reset.

Each option bit has also its complement in the same double-word. During option loading, a verification of the option bit and its complement allows the correct loading to be checked.

During option byte loading, the options are read by double-word. ECC on option words is not taken into account during OBL but only during direct software read of option area.

If the word and its complement are matching, the option word/byte is copied into the option register.

If the comparison between the word and its complement fails, a status bit OPTVERR is set. Mismatch values are forced into the option registers as follows:

- For USR OPT option, the value of mismatch is all options at ‘1’, except for BOR_LEV that is “000” (lowest threshold).
- For WRP option, the value of mismatch is the default value “No protection”.
- For RDP option, the value of mismatch is the default value “Level 1”.
- For PCROP, the value of mismatch is “all memory protected”.
- For BOOT_LOCK option, the value of mismatch is “CPU boot lock disabled”
- For OPTVAL option, the value of mismatch is “not valid”. OPTVAL is a check word programmed at the last user option address. It is used to check if all user options have been programmed during an OPTSTART. If the user option program sequence has not terminated completely, OPTVAL is not be programmed and OPTNV is set.

If the OPTVAL option indicates “not valid”, SRAM1, SRAM2, and PKA SRAM memories are erased.

Table 10. Option loading control

OPTVERR	OPTNV	Description
0	0	Options correctly loaded and OPTVAL is “Valid”. Device is non-secure.
0	1	Does not occur
1	0	OPTVAL option is correctly loaded as “Valid” but some or all other options and engineering bits are corrupted: mismatch values are loaded. – When secure option is loaded correctly, security is applied according to the loaded secure option values. – When secure option is corrupted, security is applied on the full memory as indicated by the loaded mismatch value.
1	1	Some or all options and engineering bits are corrupted: mismatch values are loaded. OPTVAL is correctly loaded as “not Valid”. Security applied on full memories irrespective of the loaded secure option values.

On system reset rising, internal option registers are copied into the following option registers that can be read and written by software:

- FLASH_OPTR
- FLASH_PCROP1xSR (x=A or B)
- FLASH_PCROP1xER (x=A or B)
- FLASH_WRP1xR (x=A or B)

These registers are also used to modify the options. If these registers are not modified by the user, they reflect the options states of the system. See *Modify user options* for more details.

3.5 Flash memory protection

The main flash memory can be protected against external accesses with the readout protection (RDP). The pages can also be protected against unwanted write (WRP) due to loss of program counter context. The write protection WRP granularity is 2 Kbytes.

Apart from the RDP and WRP, the flash memory can also be protected against read and write from third parties (PCROP). The PCROP granularity is 1 Kbyte.

3.5.1 Readout protection (RDP)

The readout protection is activated by setting the RDP option byte and performing an option byte programming with OPTSTRT followed by a OBL_LAUNCH, POR or wakeup from Standby or Shutdown mode. The readout protection protects the main flash memory, the option bytes, the backup registers (TAMP_BKPxR in TAMP) and SRAM2.

There is no exception while the debugger is connected.

There are three levels of readout protection from no protection (level 0) to maximum protection or no debug (level 2).

The flash memory is protected when the RDP option byte and its complement contain the pair of values shown in the table below.

Table 11. Flash memory readout protection status

RDP byte value	RDP complement value	RDP level
0xAA	0x55	Level 0
Any value except 0xAA or 0xCC	Any value (not necessarily complementary), except 0x55 and 0x33	Level 1 (default)
0xCC	0x33	Level 2

The system memory area is read accessible whatever the protection level. It is never accessible for program/erase operation.

Level 0: no protection

Read, program and erase operations into the main flash memory area are possible. The option bytes, SRAM2 and backup registers are also accessible by all operations.

Level 1: readout protection

This is the default protection level when the RDP option byte is erased. It is defined as well when the RDP value is at any value different from 0xAA and 0xCC, or even if the complement is not correct.

- User mode

  The code executing in user mode (Boot flash) can access the main flash memory, option bytes, SRAM2 and backup registers with all operations.
- Debug, boot RAM and bootloader and SFI/RSS modes

  In debug mode or when the code is running from boot RAM or bootloader or SFI/RSS, the main flash memory, backup registers (RTC_BKPxR in the RTC) and SRAM2 are totally inaccessible. In these modes, a read or write access to the flash memory generates a bus error and a hard fault interrupt.

Caution: In case the level 1 is configured and no PCROP areas are defined, it is mandatory to set PCROP_RDP bit to 1 (full mass erase when the RDP level is decreased from level 1 to level 0). In case the level 1 is configured and a PCROP area is defined, if the user code needs to be protected by RDP but not by PCROP, it must not be placed in a page containing a PCROP area.

Level 2: no debug

In this level, the protection level 1 is guaranteed. In addition, the CPU debug port, the boot from RAM (boot RAM mode) and the boot from system memory (bootloader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the main flash memory. On the contrary, only read and secure write operations can be performed on the option bytes.

Note: The debug feature is also disabled under reset.

STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set.

Change the readout protection level

It is easy to move from level 0 to level 1 by changing the value of the RDP byte to any value (except 0xCC). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 directly from level 0 or from level 1. Once in level 2 it is no more possible to modify the readout protection level.

When the RDP is reprogrammed to the value 0xAA to move from level 1 to level 0, a mass erase of the main flash memory is performed if PCROP_RDP is set in FLASH_PCROP1AER. Backup registers (RTC_BKPxR in the RTC), SRAM1, SRAM2 and PKA SRAM are also erased. The user options except PCROP protection are set to their previous values copied from FLASH_OPTR, FLASH_WRP1xR (x= A or B). PCROP is disabled. The OTP area is not affected by mass erase and remains unchanged.

If the bit PCROP_RDP is cleared in FLASH_PCROP1AER, the full mass erase is replaced by a partial mass erase that is successive page erases, except for the pages protected by PCROP. This is done in order to keep the PCROP code. Only when the flash memory is erased, options are re-programmed with their previous values. This is also true for FLASH_PCROP1xSR and FLASH_PCROP1xER registers (x= A or B).

Table 12. RDP regression from level 1 to level 0 and memory erase

PCROP	PCROP_RDP	Comment
None	x	Flash, SRAM1, SRAM2, PKA SRAM and backup registers mass erase
Partial	1	
	0	Flash multiple page erase of all non-PCROP pages SRAM1, SRAM2, PKA SRAM and backup registers erased (PCROP Flash pages conserved)
Complete		Flash, SRAM1, SRAM2 and backup registers conserved PKA SRAM erase.

Note: Full mass erase or partial mass erase is performed only when level 1 is active and level 0 is requested. When the protection level is increased (0→1, 1→2, 0→2), there is no mass erase.

To validate the protection level change, the option bytes must be reloaded through the OBL_LAUNCH bit in FLASH_CR, or a POR, or wakeup from Standby or Shutdown mode.

Figure 4. Changing the RDP level

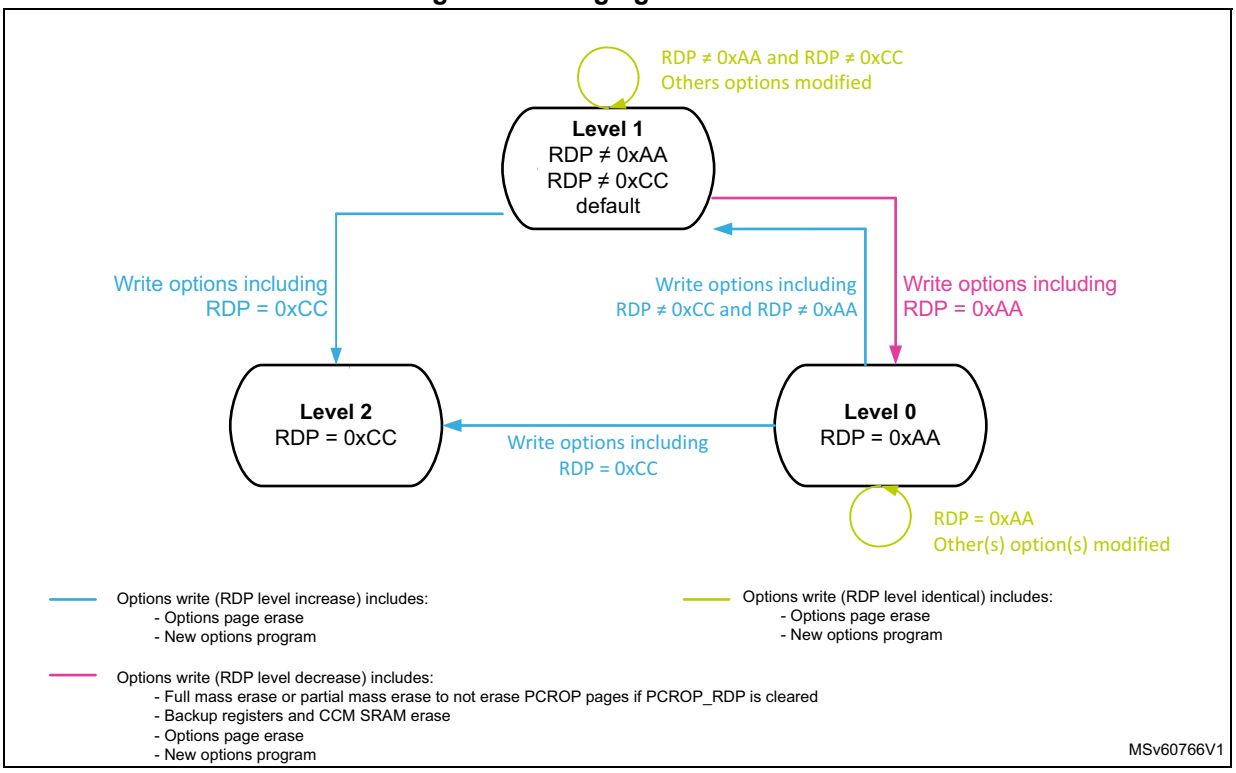


Table 13. Access status versus protection level and execution modes

Area	Protection level	User execution (BootFromFlash)			Debug/ BootFromRam/ BootFromLoader/|BootFromSFI /RSS		
		Read	Write	Erase	Read	Write	Erase
Main flash memory	1	Yes	Yes	Yes	No	No	No(4)
	2	Yes	Yes	Yes	N/A(1)	N/A(1)	N/A(1)
System memory(2)	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A(1)	N/A(1)	N/A(1)
Option bytes	1	Yes	Yes(3)	Yes	Yes	Yes(3)	Yes
	2	Yes	No	No	N/A(1)	N/A(1)	N/A(1)
Backup registers	1	Yes	Yes	N/A	No	No	No(4)
	2	Yes	Yes	N/A	N/A(1)	N/A(1)	N/A(1)
SRAM2	1	Yes	Yes	N/A	No	No	No(5)
	2	Yes	Yes	N/A	N/A(1)	N/A(1)	N/A(1)

1. When the protection level 2 is active, the debug port, the boot from RAM and the boot from system memory are disabled.
2. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
3. The Flash non secure main memory is erased when the RDP option byte is programmed from level 1 to level 0.
4. The backup registers are erased when RDP changes from level 1 to level 0.
5. SRAM1, SRAM2 and PKA SRAM are erased when RDP changes from level 1 to level 0.

3.5.2 Proprietary code readout protection (PCROP)

Two parts of the flash memory can be protected against read and write from third parties.

The protected area is execute-only: it can only be reached by the STM32 CPU with an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. The PCROP areas have a 1-Kbyte granularity. An additional option bit (PCROP_RDP) defines if the PCROP area is erased or not when the RDP protection is changed from level 1 to level 0 (refer to *Change the readout protection level*).

Each PCROP area is defined by a start page offset and an end page offset into the flash memory. These offsets are defined in the PCROP address registers (FLASH_PCROP1ASR, FLASH_PCROP1AER), FLASH_PCROP1BSR and FLASH_PCROP1BER).

A PCROP area is defined from the address:

Flash memory base address + [PCROP1x_STRT x 0x400] (included) to the address

Flash memory base address + [(PCROP1x_END+1) x 0x400] (excluded).

The minimum PCROP area size is two PCROP pages (2 Kbytes)
PCROP1x_END = PCROP1x_STRT + 1.

When PCROP1x_END = PCROP1x_STRT, the full flash memory is PCROP protected.

For example, to protect by PCROP from the address 0x0801 2F80 (included) to the address 0x0801 D004 (included), if boot in flash is selected, one of the FLASH_PCROP1xSR and FLASH_PCROP1xER registers (x =A or B) must be programmed with:

- PCROP1x_STRT = 0x4B (PCROP area first address 0x0801 2C00)
- PCROP1x_END = 0x74 (PCROP area last address 0x0801 D3FF)

Any data read access performed through a PCROP protected area triggers the RDERR flag error.

Any PCROP protected address is also write protected and any write access to one of these addresses triggers WRPERR.

Any PCROP area is also erase protected. Consequently, any erase to a page in this zone is impossible (including the page containing the start address and the end address of this zone). Moreover, a software mass erase cannot be performed if one zone is PCROP protected.

In the previous example, due to erase by page, all pages from page 0x4B to 0x74 are protected. In case of page erase, all addresses from 0x0801 2C00 to 0x0801 D3FF cannot be erased.

Deactivation of PCROP can only occurs when the RDP is changing from level 1 to level 0. If the user options modification tries to clear PCROP or to decrease the PCROP areas, the options programming is launched but PCROP areas stays unchanged. On the contrary, it is possible to increase the PCROP areas.

When option bit PCROP_RDP is cleared and when the RDP is changing from level 1 to level 0, the full mass erase is replaced by a partial mass erase to preserve the PCROP area (refer to *Change the readout protection level*). In this case, PCROP1x_STRT and PCROP1x_END (x =A or B) are not erased.

Table 14: PCROP protection

PCROP registers values (x = A or B)	PCROP protection area
PCROP1x_STRT = PCROP1x_END	No PCROP1x, unprotected
PCROP1x_STRT > PCROP1x_END	No PCROP1x, unprotected
PCROP1x_STRT < PCROP1x_END	Pages from PCROP1x_STRT to PCROP1x_END are protected

Note: It is recommended to align PCROP areas with the page granularity when using PCROP_RDP, or to leave free the rest of the page where PCROP zones starts or ends.

3.5.3 Write protection (WRP)

The user area in the flash memory can be protected against unwanted write operations. Two write-protected (WRP) areas can be defined, with 2-Kbyte granularity page. Each area is defined by a start page offset and an end page offset related to the physical flash memory base address. These offsets are defined in the WRP address registers FLASH_WRP1AR and FLASH_WRP1BR.

The WRP “x” area (x=A, B) is defined from the address:

Flash memory Base address + [WRP1x_STRT x 0x800] (included) to the address:

Flash memory Base address + [(WRP1x_END+1) x 0x800] (excluded).

The minimum WRP area size is one WRP 2_Kbyte page, WRP1x_END = WRP1x_STRT.

For example, to protect by WRP from the address 0x0801 2000 (included) to the address 0x0801 9FFF (included), if boot in flash is selected, FLASH_WRP1AR register must be programmed with:

- WRP1A_STRT = 0x24
- WRP1A_END = 0x33

WRP1B_STRT and WRP1B_END in FLASH_WRP1BR can be used instead (area “B” in flash memory).

When WRP is active, it cannot be erased or programmed. Consequently, a software mass erase cannot be performed if one area is write-protected.

If an erase/program operation to a write-protected part of the flash memory is attempted, the write protection error flag (WRPERR) is set in FLASH_SR. This flag is also set for any write access to on the following area:

- OTP area
- part of the flash memory that can never be written like the ICP
- PCROP area

Note: When the flash memory readout protection level is selected (RDP level1), it is not possible to program or erase the memory if the CPU debug features are connected (JTAG or single wire) or boot code is being executed from RAM or system flash, even if WRP is not activated. Any attempt generates an hard fault (BusFault).

Table 15: WRP protection

WRPx registers values (x = A or B)	WRP protection area
WRP1x_STRT = WRP1x_END	Page WRP1x is protected
WRP1x_STRT > WRP1x_END	No WRP, unprotected
WRP1x_STRT < WRP1x_END	Pages from WRP1x_STRT to WRP1x_END are protected

Note: To validate the WRP options, the option bytes must be reloaded through the OBL_LAUNCH bit in FLASH_CR.

3.5.4 Security (ESE)

When OPTVAL option indicates “not valid” or the comparison between the user options word and its complement fails (indicated in OPTVERR), the Flash memory is secured and SRAM1, SRAM2 and PKA SRAM memories are erased. The CPU is no longer able to execute firmware from the Flash memory. When booting in SRAM1, the CPU debug is enabled. In this case, any CPU firmware can be downloaded in SRAM or JTAG can be used to perform an RDP regression from level1 to level 0 with ESE = 0.This restores valid option bytes and erases all user Flash memory, SRAM1, SRAM2 and PKA SRAM content.

3.5.5 CPU boot lock chain of trust

The BOOT_LOCK forces the CPU to boot from the user flash memory, regardless of what is selected by BOOT0 and BOOT1. When BOOT_LOCK is enabled and BOOT0/BOOT1 select anything different than the user flash memory boot, the system boots anyway from the user flash memory. System boot via BOOT0/BOOT1 from SRAM1 or bootloader is no longer possible.

It is still possible to boot the CPU according to the software selected remap by MEM_MODE bits in *SYSCFG memory remap register (SYSCFG_MEMRMP)* from SRAM1 or bootloader.

3.6 FLASH program erase suspension

Flash program and erase operation can be suspended by setting the PES bit in FLASH_ACR. This feature is useful when executing time critical sections by a CPU. It makes possible to suspend any new program or erase operation from being started, preventing CPU instruction and data fetches from being blocked.

When PES bit is set, the following occurs:

- Any ongoing program or erase operation is completed.

  The maximum latency for a flash program erase suspension is the maximum time for one program or erase operation to complete (see product datasheets for more information on the flash program and erase timing).
- All new requested program and erase operations is not started but suspended.

  PESD bit in FLASH_SRis setas soon as PES is set, no matter if a program/erase is currently suspended. This allows a CPU to test PESD prior to requesting a program or an erase operation.

When PES bit is reset to 0, a suspended program or erase operation is started: the PESD bit is cleared.

3.7 FLASH interrupts

Table 16. Flash interrupt requests

Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit
End of operation	EOP(1)	Write EOP=1	EOPIE
Operation error	OPERR(2)	Write OPERR=1	ERRIE
Readout protection error	RDERR	Write RDERR=1	RDERRIE
Write protection error	WRPERR	Write WRPERR=1	N/A
Size error	SIZERR	Write SIZERR=1	N/A
Programming sequential error	PROGERR	Write PROGERR=1	N/A
Programming alignment error	PGAERR	Write PGAERR=1	N/A
Programming sequence error	PGSERR	Write PGSERR=1	N/A
Data miss during fast programming error	MISSERR	Write MISSERR=1	N/A
Fast programming error	FASTERR	Write FASTERR=1	N/A
ECC error correction	ECCC	Write ECCC=1	ECCCIE
ECC double-error (NMI)	ECCD	Write ECCD=1	N/A

1. EOP is set only if EOPIE is set.
2. OPERR is set only if ERRIE is set.

3.8 FLASH registers

3.8.1 FLASH access control register (FLASH_ACR)

Address offset: 0x000

Reset value: 0x0000 0600

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PES	Res.	Res.	DCRST	ICRST	DCEN	ICEN	PRFTEN	Res.	Res.	Res.	Res.	Res.	LATENCY[2:0]		
rw			rw	rw	rw	rw	rw						rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **EMPTY**: Flash user area empty

When read, this bit indicates whether the first location of the user flash is erased or has a programmed value.

0: Read: user flash programmed

1: Read: user flash empty

Bit 15 **PES**: CPU program/erase suspend request

0: Flash program and erase operations granted

1: Any new flash program and erase operation is suspended until this bit is cleared. The PESD bit in FLASH_SR is set when PES bit in FLASH_ACRis set.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **DCRST**: CPU data cache reset

0: CPU data cache not reset

1: CPU data cache reset

This bit can be written only when the data cache is disabled.

Bit 11 **ICRST**: CPU instruction cache reset

0: CPU instruction cache not reset

1: CPU instruction cache reset

This bit can be written only when the instruction cache is disabled.

Bit 10 **DCEN**: CPU data cache enable

0: CPU data cache disabled

1: CPU data cache enabled

Bit 9 **ICEN**: CPU instruction cache enable

0: CPU instruction cache disabled

1: CPU instruction cache enabled

Bit 8 **PRFTEN:** CPU prefetch enable

0: CPU prefetch disabled

1: CPU prefetch enabled

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **LATENCY[2:0]**: Latency

These bits represent the ratio of the flash HCLK clock period to the flash memory access time.

000: Zero wait state

001: One wait state

010: Two wait states

Others: reserved

3.8.2 FLASH key register (FLASH_KEYR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Flash key

The following values must be written consecutively to unlock FLASH_CR, thus enabling programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

3.8.3 FLASH option key register (FLASH_OPTKEYR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTKEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OPTKEY[31:0]:** Option byte key lower bits

The following values must be written consecutively to unlock the flash memory option registers, enabling option byte programming/erasing operations:
KEY1: 0x0819 2A3B
KEY2: 0x4C5D 6E7F

3.8.4 FLASH status register (FLASH_SR)

Address offset: 0x010

Reset value: 0x000X 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PESD	CFGBSY	Res.	BSY
												r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTVERR	RDERR	OPTNV	Res.	Res.	Res.	FASTERR	MISSERR	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
rc_w1	rc_w1	r				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **PESD**: Program/erase operation suspended

This bit is set and reset by hardware.
Set when PES bit in FLASH_ACR is set.
Cleared when PES bitin FLASH_ACRis cleared.
When set, new program or erase operations are not started.

Bit 18 **CFGBSY**: Program or erase configuration busy

This bit is set and reset by hardware (set when first word is sent and reset when program operation completes or is interrupted by an error).
When set, the program and erase settings in PG, PNB[6:0], PER, and MER bits in FLASH_CR are used (busy) and cannot be changed (a programming or erase operation is ongoing).
When reset, the program and erase settings in PG, PNB[6:0], PER, and MER bits in FLASH_CR can be modified.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **BSY**: Busy

This bit indicates that a flash operation requested in FLASH_CR is in progress. This bit is set at the beginning of a flash operation and reset when the operation finishes or when an error occurs.

Bit 15 **OPTVERR**: Option and engineering bits loading validity error

Set by hardware when the options and engineering bits read may not be the one configured by the user or production. If options and engineering bits are not properly loaded, OPTVERR is set again after each system reset. Option bytes that fail loading are forced to a safe value (see *Section 3.4.2: Option bytes programming*).
This bit is cleared by writing 1.

Bit 14 **RDERR**: PCROP read error

Set by hardware when an address to be read through the D-bus belongs to a read protected area of the flash memory (PCROP protection). An interrupt is generated if RDERRIE is set in FLASH_CR.
This bit is cleared by writing 1.

Bit 13 **OPTNV**: User option OPTVAL indication

This bit is set and reset by hardware.

0: The OBL user option OPTVAL indicates “valid” (user option program sequence has not terminated completely).

1: The OBL user option OPTVAL indicates “not valid” (OPTVAL check word has been erroneously read).

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **FASTERR**: Fast programming error

Set by hardware when a fast programming sequence (activated by FSTPG) is interrupted due to an error (alignment, size, write protection or data miss). The corresponding status bit (PGAERR, SIZERR, WRPERR or MISSERR) is set at the same time.

This bit is cleared by writing 1.

Bit 8 **MISSERR**: Fast programming data miss error

In Fast programming mode, 32 double-words (256 bytes) must be sent to the flash memory successively and the new data must be sent to the logic control before the current data is fully programmed.

This bit is set by hardware when the new data is not present in time and cleared by writing 1.

Bit 7 **PGSERR**: Programming sequence error

This bit is set by hardware when a write access to the flash memory is performed by the code, while PG or FSTPG have not been set previously. This bit is also set by hardware when PROGERR, SIZERR, PGAERR, WRPERR, MISSERR or FASTERR is set due to a previous programming error.

This bit is cleared by writing 1.

Bit 6 **SIZERR**: Size error

This bit is set by hardware when the size of the access is a byte or half-word during a program or a fast program sequence. Only double-word programming is allowed (consequently: word access).

This bit is cleared by writing 1.

Bit 5 **PGAERR**: Programming alignment error

This bit is set by hardware when the data to program cannot be contained in the same double-word (64 bits) flash memory in case of standard programming, or if there is a change of page during fast programming.

This bit is cleared by writing 1.

Bit 4 **WRPERR**: Write protection error

This bit is set by hardware when an address to be erased/programmed belongs to a write-protected part (by WRP, PCROP or RDP level 1) of the flash memory.

This bit is cleared by writing 1.

Bit 3 **PROGERR**: Programming error

This bit is set by hardware when a double-word address to be programmed contains a value different from 0xFFFF FFFF FFFF FFFF before programming, except if the data to write is 0x0000 0000 0000 0000.

This bit is cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Operation error

This bit is set by hardware when a flash memory operation (program/erase) completes unsuccessfully. This bit is set only if error interrupts are enabled (ERRIE = 1).
This bit is cleared by writing 1.

Bit 0 **EOP**: End of operation

This bit is set by hardware when one or more flash memory operation (program/erase) completes successfully. This bit is set only if the end of operation interrupts are enabled (EOPIE = 1).
This bit is cleared by writing 1.

3.8.5 FLASH control register (FLASH_CR)

Address offset: 0x014

Reset value: 0xC000 0000

Access: no wait state when no flash memory operation is ongoing. Word, half-word and byte access.

This register cannot be modified when CFGBSY is set in FLASH_SR.

When PESD is cleared in FLASH_SR, the register write access is stalled until the CFGBSY bit is cleared.

When PESD is set in FLASH_SR and a program or an erase operation is ongoing, the register write access causes a bus error.

When PESD is set in FLASH_SR but there is no ongoing programming or erase operation, the register write access is completed, but the requested operation is suspended. BSY/CFGBSY is set and remains 1 until suspend is deactivated by clearing the PES bits in FALSH_ACR. Consequently PESD goes back to 0 and the suspended operation completes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	RDERRIE	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPTSTRT	STRT
rs	rs			rc_w1	rw	rw	rw						rw	rs	rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	PNB[6:0]							MER	PER	PG
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** FLASH_CR lock

This bit can only be set by software. When set, the FLASH_CR register is locked. This bit is cleared by hardware after detecting the unlock sequence.

In case of an unsuccessful unlock operation, this bit remains set until the next system reset.

Bit 30 **OPTLOCK:** Options lock

This bit can only be set by software. When set, all bits concerning user option in FLASH_CR and so option page are locked. This bit is cleared by hardware after detecting the option unlock sequence. The LOCK bit must be cleared before doing the unlock sequence for OPTLOCK bit.

In case of an unsuccessful option unlock operation, this bit remains set until the next reset.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **OBL_LAUNCH**: forces the option byte loading

When set to 1, this bit forces the option byte reloading. This bit is cleared only when the option byte loading is complete. It cannot be written if OPTLOCK is set.

0: Option byte loading completed

1: Option byte loading requested

Bit 26 **RDERRIE**: PCROP read error interrupt enable

This bit enables the interrupt generation when the RDERR bit in FLASH_SR is set to 1.

0: PCROP read error interrupt disabled

1: PCROP read error interrupt enabled

Bit 25 **ERRIE**: error interrupt enable

This bit enables the interrupt generation when the OPERR bit in FLASH_SR is set to 1.

0: OPERR error interrupt disabled

1: OPERR error interrupt enabled

Bit 24 **EOPIE**: end of operation interrupt enable

This bit enables the interrupt generation when the EOP bit in FLASH_SR is set to 1.

0: EOP Interrupt disabled

1: EOP Interrupt enabled

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **FSTPG**: fast programming

0: Fast programming disabled

1: Fast programming enabled

Bit 17 **OPTSTRT**: options modification start

When set, this bit triggers an options programming operation. When RDP level is regressed from level 1 to level 0, this bit also launches a flash memory, SRAM1 and SRAM2 erase.

This bit is set only by software and cleared when BSY is cleared in FLASH_SR.

Bit 16 **STRT**: start

When set, this bit triggers an erase operation. If MER and PER are reset and STRT is set, an unpredictable behavior may occur without generating any error flag. This condition is forbidden.

This bit is set only by software and cleared when BSY is cleared in FLASH_SR.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: page number selection
These bits select the 2-Kbyte page to erase.
0x00: page 0
0x01: page 1
...
0x3F: page 63

Bit 2 **MER**: mass erase
When set, this bit triggers the mass erase (all user pages).

Bit 1 **PER**: page erase
0: page erase disabled
1: page erase enabled

Bit 0 **PG**: programming
0: Flash programming disabled
1: Flash programming enabled

3.8.6 FLASH ECC register (FLASH_ECCR)

Address offset: 0x018

Reset value: 0x0000 0000

Access: no wait state when no flash memory operation is ongoing. Word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCCIE	Res.	Res.	Res.	SYSF_ECC	Res.	Res.	Res.	ADDR_ECC[16]
rc_w1	rc_w1						rw				r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ECCD:** ECC detection
Set by hardware when two ECC errors have been detected. When this bit is set, a NMI is generated.
This bit is cleared by writing 1.

Bit 30 **ECCC**: ECC correction
Set by hardware when one ECC error has been detected and corrected. An interrupt is generated if ECCIE is set.
This bit is cleared by writing 1.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **ECCCIE**: ECC correction interrupt enable
0: ECCC interrupt disabled
1: ECCC interrupt enabled

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **SYSF_ECC**: system flash memory ECC fail

This bit indicates that the ECC error correction or double ECC error detection is located in the system flash memory.

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:0 **ADDR_ECC[16:0]**: ECC fail double-word address

This bit indicates that double-word address is concerned by the ECC error correction or causes the double ECC error detection.

3.8.7 FLASH option register (FLASH_OPTR)

Address offset: 0x020

Reset value: 0x3FFF F0AA

Default reset value from ST production is given. Subsequently, 0bXX11 XXXX X111 XXXX 1XXX XXXX XXXX XXXX, the option bits are loaded with user values from the flash memory at reset release.

Access: no wait state when no flash memory operation is ongoing. Word, half-word and byte access.

This register can only be written by the CPU in RDP level 0 or RDP level 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	BOOT_LOCK	Res.	Res.	nBOOT0	nSWBOOT0	SRAM_RST	SRAM2_PE	nBOOT1	Res.	Res.	Res.	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW
	rw			rw	rw	rw	rw	rw				rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	BOR_LEV[2:0]			ESE	RDP[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **BOOT_LOCK:** CPU boot lock enable option bit

This bit may be set by software at any time, but a write to clear is only taken into account in one of the following conditions:

- when staying in RDP level 0
- when regressing RDP level from 1 to 0

0: CPU boot lock disabled

1: CPU boot lock enabled

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **nBOOT0:** nBOOT0 option bit

If nSWBOOT0 bit selects BOOT0 to be taken from option bit nBOOT0, then this bit, together with option nBOOT1, selects the boot modes (from the user flash memory, SRAM1 or system flash memory). Refer to *Section 2.2: Boot configuration*.

0: nBOOT0=0

1: nBOOT0=1

Bit 26 **nSWBOOT0:** software BOOT0 selection
0: BOOT0 taken from the option bit nBOOT0
1: BOOT0 taken from PH3/BOOT0 pin

Bit 25 **SRAM_RST:** SRAM1 and SRAM2 erase when system reset
0: SRAM1 and SRAM2 erased when a system reset occurs
1: SRAM1 and SRAM2 not erased when a system reset occurs
Note: PKA SRAM is always erased on any system.

Bit 24 **SRAM2_PE:** SRAM2 parity check enable
0: SRAM2 parity check enabled
1: SRAM2 parity check disable

Bit 23 **nBOOT1:** boot configuration
Together with the BOOT0 pin or option bit nBOOT0 (depending on nSWBOOT0 option bit configuration), this bit selects boot mode from the user flash memory, SRAM or system flash memory. Refer to *Section 2.2: Boot configuration*.

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 **WWDG_SW:** window watchdog selection
0: Hardware window watchdog
1: Software window watchdog

Bit 18 **IWDG_STDBY:** independent watchdog counter freeze in Standby mode
0: Independent watchdog counter frozen in Standby mode
1: Independent watchdog counter running in Standby mode

Bit 17 **IWDG_STOP:** independent watchdog counter freeze in Stop mode
0: Independent watchdog counter frozen in Stop mode
1: Independent watchdog counter running in Stop mode

Bit 16 **IWDG_SW:** independent watchdog selection
0: Hardware independent watchdog
1: Software independent watchdog

Bit 15 Reserved, must be kept at reset value.

Bit 14 **nRST_SHDW:** reset generation in Shutdown mode
0: Reset generated when entering the Shutdown mode
1: No reset generated when entering the Shutdown mode

Bit 13 **nRST_STDBY:** reset generation in Standby mode
0: Reset generated when entering the Standby mode
1: No reset generated when entering the Standby mode

Bit 12 **nRST_STOP:** reset generation in Stop mode
0: Reset generated when entering the Stop mode
1: No reset generated when entering the Stop mode

Bits 11:9 **BOR_LEV[2:0]:** BOR reset Level
These bits contain the V_{DD} supply level threshold that activates/releases the reset.
000: BOR level 0. Reset level threshold is around 1.7 V
001: BOR level 1. Reset level threshold is around 2.0 V
010: BOR level 2. Reset level threshold is around 2.2 V
011: BOR level 3. Reset level threshold is around 2.5 V
100: BOR level 4. Reset level threshold is around 2.8 V

Bit 8 **ESE:** system security enable flag
When read, this bit indicates whether the system security is enabled, meaning user option FSD = 0. Writing 0 to this bit and regressing the RDP from level 1 to level 0 disables the security
0: Security disabled
1: Security enabled

Bits 7:0 **RDP[7:0]:** readout protection level
0xAA: Level 0, readout protection not active
0xCC: Level 2, chip readout protection active
Others: Level 1, memories readout protection active

Note: Take care about PCROP_RDP configuration in level 1. Refer to Level 1: readout protection for more details.

3.8.8 FLASH PCROP zone A start address register (FLASH_PCROP1ASR)

Address offset: 0x024

Reset value: 0xFFFF FFFF

Default reset value from ST production is given. Subsequently, 0b1111 1111 1111 1111 1111 1111 XXXX XXXX, the option bits are loaded with user values from the flash memory at reset release.

Access: no wait state when no flash memory operation is ongoing. Word, half-word access.

This register can only be written by the CPU in RDP level 0 or RDP level 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1A_STRT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PCROP1A_STRT[7:0]:** PCROP1A area start offset
PCROP1A_STRT contains the first 1-Kbyte page of the PCROP1A area.

3.8.9 FLASH PCROP zone A end address register (FLASH_PCROP1AER)

Address offset: 0x028

Reset value: 0xFFFF FF00

Default reset value from ST production is given. Subsequently, 0bX111 1111 1111 1111 1111 1111 XXXX XXXX, the option bits are loaded with user values from flash memory at reset release.

Access: no wait state when no flash memory operation is ongoing. Word, half-word access. PCROP_RDP bit can be accessed with byte access

This register can only be written by the CPU in RDP level 0 or RDP level 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1A_END[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **PCROP_RDP:** PCROP area preserved when RDP level decreased
This bit is set only. It is reset after a full mass erase due to a change of RDP from level 1 to level 0.
0: PCROP area not erased when the RDP level is decreased from level 1 to level 0
1: PCROP area erased when the RDP level is decreased from level 1 to level 0 (full mass erase)

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **PCROP1A_END[7:0]:** PCROP1A area end offset
PCROP1A_END contains the last 1-Kbyte page of the PCROP1A area.

3.8.10 FLASH WRP area A address register (FLASH_WRP1AR)

Address offset: 0x02C

Reset value: 0xFF80 FFFF

Default reset value from ST production is given as.0b1111 1111 1XXX XXXX 1111 1111 1XXX XXXX, the option bits are loaded with user values from the flash memory at reset release.

Access: no wait state when no flash memory operation is ongoing. Word, half-word and byte access.

This register can only be written by the CPU in RDP level 0 or RDP level 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_END[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP1A_END[6:0]:** WRP area A end offset
Contains the last 2-Kbyte page of the WRP area A.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **WRP1A_STRT[6:0]:** WRP area A start offset
Contains the first 2-Kbyte page of the WRP area A.

3.8.11 FLASH WRP area B address register (FLASH_WRP1BR)

Address offset: 0x030

Reset value: 0xFF80 FFFF

Default reset value from ST production is given as 0b1111 1111 1XXX XXXX 1111 1111 1XXX XXXX, the option bits are loaded with user values from flash memory at reset release.

Access: no wait state when no flash memory operation is ongoing. Word, half-word and byte access.

This register can only be written by the CPU in RDP level 0 or RDP level 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_END[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP1B_END[6:0]:** WRP area B end offset
WRPB1_END contains the last 2-Kbyte page of the WRP area B.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **WRP1B_STRT[6:0]:** WRP area B start offset
WRPB1_END contains the first 2-Kbyte page of the WRP area B.

3.8.12 FLASH PCROP zone B start address register (FLASH_PCROP1BSR)

Address offset: 0x034

Reset value: 0xFFFF FFFF

Default reset value from ST production is given. Subsequently, 0b1111 1111 1111 1111 1111 1111 XXXX XXXX, the option bits are loaded with user values from the flash memory at reset release.

Access: no wait state when no flash memory operation is ongoing. Word and half-word access.

This register can only be written by the CPU in RDP level 0 or RDP level 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1B_STRT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PCROP1B_STRT[7:0]:** PCROP1B area start offset
Contains the first 1-Kbyte page of the PCROP1B area.

3.8.13 FLASH PCROP zone B end address register (FLASH_PCROP1BER)

Address offset: 0x038

Reset value: 0xFFFF FF00

Default reset value from ST production is given. Subsequently, 0b1111 1111 1111 1111 1111 1111 XXXX XXXX, the option bits are loaded with user values from the flash memory at reset release.

Access: no wait state when no flash memory operation is ongoing. Word and half-word access.

This register can only be written by the CPU in RDP level 0 or RDP level 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1B_END[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PCROP1B_END[7:0]:** PCROP1B area end offset
Contains the first 1-Kbyte page of the PCROP1B area.

3.8.14 FLASH register map

Table 17. Flash interface register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**FLASH_ACR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	PES	Res.	Res.	DCRST	ICRST	DCEN	ICEN	PRFTEN	Res.	Res.	Res.	Res.	Res.	LATENCY [2:0]		
	Reset value																0	0			0	0	1	1	0						0	0	0
0x004	Reserved	Reserved.																															
0x008	**FLASH_KEYR**	KEYR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**FLASH_ OPTKEYR**	OPTKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**FLASH_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PESD	CFGBSY	Res.	BSY	OPTVERR	RDERR	OPTNV	Res.	Res.	Res.	FASTERR	MISSERR	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value													X	X		X	0	0	0				0	0	0	0	0	0	0		0	0
0x014	**FLASH_CR**	LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	RDERRIE	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPTSTRT	STRT	Res.	Res.	Res.	Res.	Res.	Res.	PNB[6:0]							MER	PER	PG
	Reset value	1	1			0	0	0	0						0	0	0							0	0	0	0	0	0	0	0	0	0
0x018	**FLASH_ECCR**	ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCCIE	Res.	Res.	Res.	SYSF_ECC	Res.	Res.	Res.	ADDR_ECC[16:0]																
	Reset value	0	0						0				0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**FLASH_OPTR**	Res.	BOOT_LOCK	Res.	Res.	nBOOT0	nSWBOOT0	SRAM_RST	SRAM2_PE	nBOOT1	Res.	Res.	Res.	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW	Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	BOR_LEV [2:0]			ESE	RDP[7:0]							
	Reset value		0			1	1	1	1	1				1	1	1	1		1	1	1	0	0	0	0	1	0	1	0	1	0	1	0
0x024	**FLASH_ PCROP1ASR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1A_STRT[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x028	**FLASH_ PCROP1AER**	PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1A_END[7:0]							
	Reset value	1																								0	0	0	0	0	0	0	0
0x02C	**FLASH_ WRP1AR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_STRT[6:0]						
	Reset value										0	0	0	0	0	0	0										1	1	1	1	1	1	1
0x030	**FLASH_ WRP1BR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_STRT[6:0]						
	Reset value										0	0	0	0	0	0	0										1	1	1	1	1	1	1

Table 17. Flash interface register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x034	**FLASH_ PCROP1BSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1B_STRT[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x038	**FLASH_ PCROP1BER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1B_END[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

4 Sub-GHz radio (SUBGHZ)

4.1 Sub-GHz radio introduction

The sub-GHz radio is an ultra-low-power sub-GHz radio operating in the 150 - 960 MHz ISM band. LoRa® and (G)FSK modulation in transmit and receive, and BPSK/(G)MSK in transmit only, allow an optimal trade-off between range, data rate and power consumption. This sub-GHz radio is compliant with the LoRaWAN® specification v1.0 and radio regulations including ETSI EN 300 220, EN 300 113, EN 301 166, FCC CFR 47 part 15, 24, 90, 101 and the ARIB STD-T30, T-67, T-108.

Features related to the LoRa modulation are only available on STM32WLE5xx devices, that support LoRa.

The sub-GHz radio consists of:

- an analog front end transceiver, capable of outputting + 15 dBm maximum power on its RFO_LP pin and + 22 dBm maximum power on RFO_HP pin
- a digital modem bank providing the following modulation schemes:
  - LoRa Rx/Tx with bandwidth (BW) from 7.8 - 500 kHz, spreading factor (SF) 5 - 12, bit rate (BR) from 0.013 to 17.4 Kbit/s (real bitrate)
  - FSK and GFSK Rx/Tx with BR from 0.6 to 300 Kbit/s
  - (G)MSK Tx with BR from 0.1 to 10 Kbit/s
  - BPSK Tx with bitrate for 100 and 600 bit/s
- a digital control comprising all data processing and sub-GHz radio configuration control
- a high-speed clock generation

4.2 Sub-GHz radio main features

The main features of the sub-GHz radio are listed below:

- Half-duplex 150 - 960 MHz ISM sub-GHz radio transceiver supporting:
  - LoRa modulation
  - (G)FSK modulation
  - (G)MSK Tx modulation
  - BPSK Tx modulation
- Programmable output power up to + 22 dBm
- Low-IF architecture, mixing the RF receive signal with a signal tone located in the negative frequency

  $f_{lo} = -f_{rf} + f_{if}$ where f_{lo} is the local RF-PLL oscillator frequency, f_{rf} is the wanted receive signal and f_{if} is the intermediate frequency
- Automatic I/Q calibration to improve image rejection

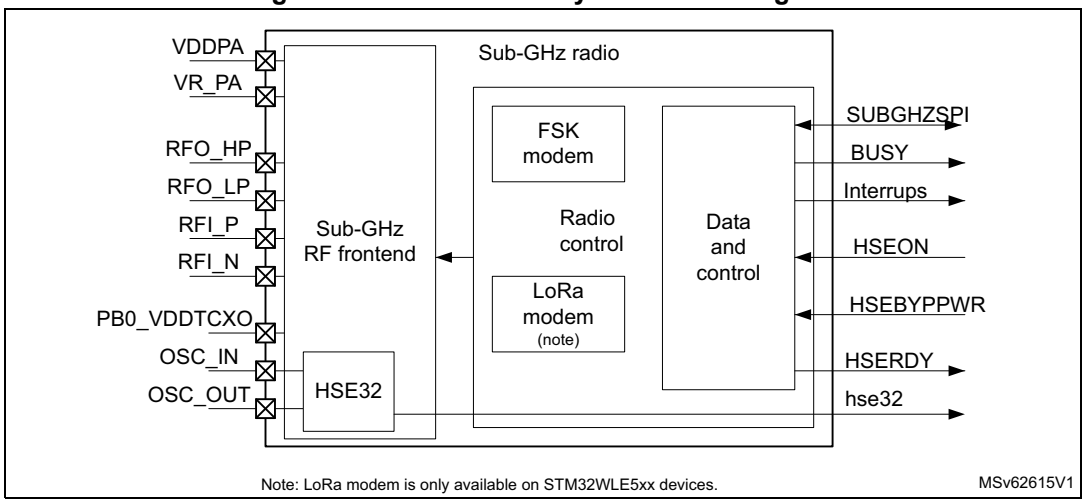
4.3 Sub-GHz radio functional description

4.3.1 General description

The sub-GHz radio provides an internal processing unit to handle communication with the system CPU. Communication is handled by commands sent over the SPI interface, and a set of interrupts is used to signal events. BUSY information signals operation activity and is used to indicate when the sub-GHz radio commands cannot be received.

The block diagram of the sub-GHz radio system is shown in the figure below.

Figure 5. Sub-GHz radio system block diagram



4.3.2 Sub-GHz radio signals

The table below gives the list of sub-GHz radio signals.

Table 18. Sub-GHz internal input/output signals

Signal name	Signal type	Description
RFO_HP	RF output	Transmit high-power PA output
RFO_LP	RF output	Transmit low-power PA output
RFI_P	RF input	Receiver differential P input
RFI_N	RF input	Receiver differential N input
OSC_IN	Analog input	HSE32 oscillator input
OSC_OUT	Analog output	HSE32 oscillator output
VDDPA	Supply	Input supply for PA regulator
VR_PA	Supply	Regulated PA supply output
PB0_VDDTCXO	Supply	Regulated TCXO supply output
hse32	Digital output	HSE32 clock signal to CPU
HSEON	Digital input	Enable HSE32 clock for CPU usage
HSEBYPPWR	Digital input	Enable VDDTCXO regulator control

Table 18. Sub-GHz internal input/output signals (continued)

Signal name	Signal type	Description
HSERDY	Digital output	HSE32 clock ready indication
SUBGHZSPI	Digital in/output	Sub-GHz radio SPI interface
BUSY	Digital output	BUSY signal
Interrupts	Digital output	IRQ interrupts

4.3.3 Transmitter

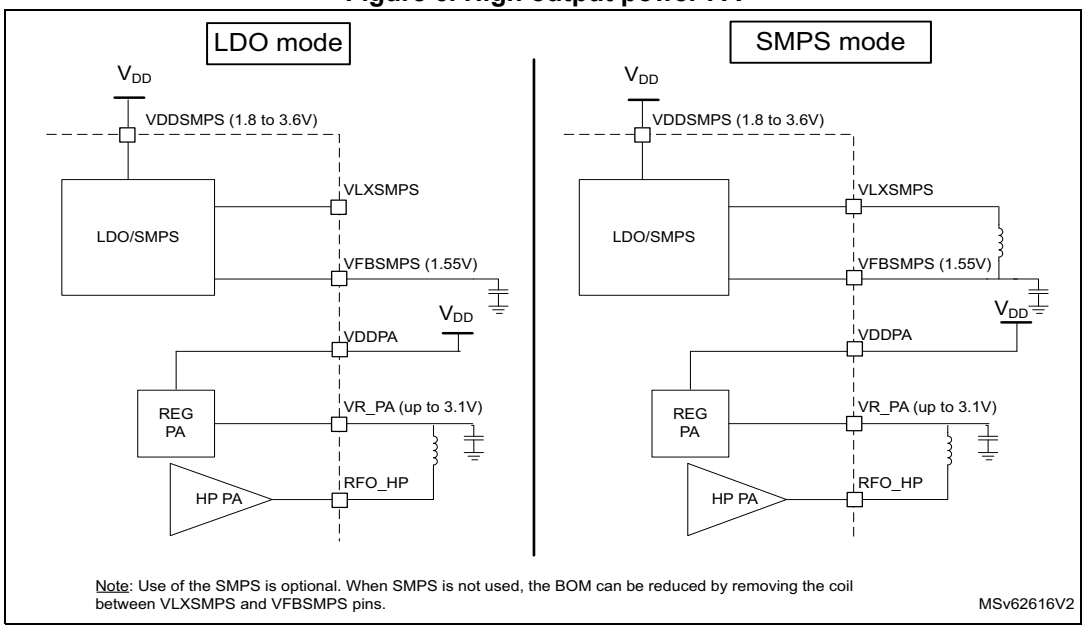
The transmit chain comprises the modulated output from the modem, that directly modulates the RF-PLL. An optional pre-filtering of the bit stream can be enabled to reduce the power in the adjacent channel also dependent on the selected modulation scheme. The modulated signal from the RF-PLL directly drives the high-power PA (HP PA) or low-power PA (LP PA).

Transmitter high output power

Transmit high output power up to + 22 dBm, is supported through the RFO_HP RF pin. The HP PA can be supplied from the PA regulator (REG PA) up to 3.1 V.

For this, the REG PA must be supplied directly from V_{DD} (on VDDSMPS pin), as shown in the figure below.

Figure 6. High output power PA



The table below gives the maximum transmit output power versus the V_{DDPA} supply level.

Table 19. Sub-GHz radio transmit high output power

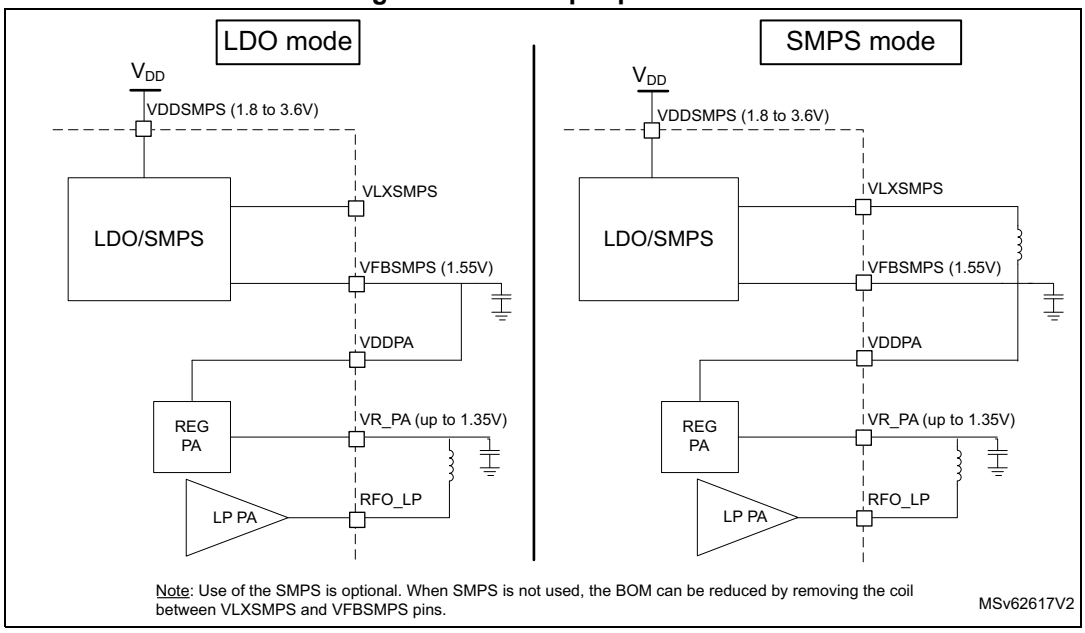
V_{DDPA} supply (V)	Transmit output power (dBm)
3.3	+ 22
2.7	+ 20
2.0	+ 16

Transmitter low output power

The transmit low output power up to + 15 dBm, is supported through the RFO_LP pin. The LP PA can be supplied from the PA regulator (REG PA) up to 1.35 V. For this, the REG PA must be supplied from the regulated V_{FBSMPS} supply at 1.55 V, as shown in the figure below.

The output power range is programmable in 32 steps of ~1 dB. The power amplifier ramping timing is also programmable.This allows adaptation to meet radio regulation requirements.

Figure 7. Low output power PA



4.3.4 Receiver

The receive chain comprises a differential low-noise amplifier (LNA), a down-converter to low-IF by mixer operation in quadrature configuration. The I and Q signals are low pass filtered and a ΣΔ ADC converts them into the digital domain. In the digital modem, the signals are decimated, further down converted and channel filtered. The demodulation is done according to the selected modulation scheme.

The down mixing to low-IF is done by mixing the receive signal with the local RF-PLL located in the negative frequency, where $-f_{lo} = -f_{rf} + f_{if}$. (where f_{lo} is the local RF-PLL frequency, f_{rf} is the received signal and f_{if} is the intermediate frequency). The wanted signal is located at $f_{rf} = f_{lo} + f_{if}$.

The receiver features automatic I and Q calibration, that improves image rejection. The calibration is done automatically at startup before using the receiver, and can be requested by command (see *Image calibration for specific frequency bands* for more details).

The receiver supports LoRa and (G)FSK modulations.

4.3.5 RF-PLL

The RF-PLL is used as the frequency synthesizer for the generation of the local oscillator frequency (f_{lo}) for both transmit and receive chains. The RF-PLL uses auto calibration and uses the 32 MHz HSE32 reference. The sub-GHz radio covers all continuous frequencies in the range between 150 to 960 MHz.

4.3.6 Intermediate frequencies

The sub-GHz radio receiver operates mostly in low-IF configuration, except for specific high-bandwidth settings.

Table 20. FSK mode intermediate frequencies

Setting name	Bandwidth (kHz)	f_{if} (kHz)
RX_BW_467	467.0	250
RX_BW_234	234.3	
RX_BW_117	117.3	
RX_BW_58	58.6	
RX_BW_29	29.3	
RX_BW_14	14.6	
RX_BW_7	7.3	
RX_BW_373	373.6	200
RX_BW_187	187.2	
RX_BW_93	93.8	
RX_BW_46	46.9	
RX_BW_23	23.4	
RX_BW_11	11.7	
RX_BW_5	5.8	
RX_BW_312	312.0	167
RX_BW_156	156.2	
RX_BW_78	78.2	
RX_BW_39	39.0	
RX_BW_19	19.5	
RX_BW_9	9.7	
RX_BW_4	4.8	

Table 21. LoRa mode intermediate frequencies

Setting name	Bandwidth [kHz]	f_{if} [kHz]
LORA_BW_500	500	0
LORA_BW_250	250	250
LORA_BW_125	125	
LORA_BW_62	62.5	
LORA_BW_41	41.67	167
LORA_BW_31	31.25	250
LORA_BW_20	20.83	167
LORA_BW_15	15.63	250
LORA_BW_10	10.42	167
LORA_BW_7	7.81	250

4.4 Sub-GHz radio clocks

4.4.1 Internal oscillators

The following sub-GHz radio dedicated internal RC oscillators are available:

- 64 kHz RC oscillator
  - optionally used during the sub-GHz radio Sleep mode to wake up the transceiver when performing periodic or duty cycled operations
  - used by the sub-GHz radio RTC for time based events
- 13 MHz RC oscillator, enabled for all sub-GHz radio SPI communication

The frequency of each sub-GHz radio internal oscillators is calibrated using the HSE32 clock at every sub-GHz radio transition from Deep-Sleep or Sleep-to-Standby, and after a sub-GHz radio reset. The calibration can also be done on demand by the command `Calibrate()`.

4.4.2 HSE32 reference clock

The high-precision 32 MHz frequency needed for the sub-GHz radio transmission and reception is taken from HSE32. The HSE32 clock can also be used by the MCU. The use of an external crystal (XTAL) or a temperature compensated crystal oscillator (TCXO) are supported. The used clock source is configured in the RCC (see *Section 6.2.1: HSE32 clock with trimming* for more details).

When using the HSE32 with a XTAL, the load capacitors are provided by the integrated capacitor banks that can be trimmed. The trimming is provided by SUBGHZ_HSEINTRIMR and SUBGHZ_HSEOUTRIMR registers. The load capacitances on OSC_IN and OSC_OUT can be trimmed separately. Software trimming must be applied after the sub-GHz radio entered Standby with HSE32 mode.

The TCXO regulator, integrated in the sub-GHz radio, can be used to supply an external temperature compensated crystal oscillator (TCXO). The regulated V_{DDTCXO} supply level is controlled through `set_TcxoMode()` command.

The sub-GHz radio, depending on the transmit output power (max + 22 dBm), can heat up the device. The heating depends on the used transmit output power and the device package. Careful PCB design using thermal heat dissipation techniques must be applied to avoid heat transfer to the HSE32 reference clock source. For the HSE32 frequency drift requirements related to the sub-GHz radio, see *Section 4.5.1: LoRa modem*.

4.5 Sub-GHz radio modems

The following modems are provided enabling the respective modulation and associated framing.

- LoRa with the LoRa framing
- FSK and MSK with a generic framing
- BPSK with a BPSK framing

The modems and frame types are set by `Set_PacketType()` command. The current used modem and frame type can be obtained by `Get_PacketType()` command.

Once the modem and frame to be used are selected, modulation and packet parameters can be defined by `Set_ModulationParams()` and `Set_PacketParams()` commands.

The LoRa and (G)FSK[(a)] modems support both transmission and reception.

The (G)MSK and BPSK modems only supports transmission.

The framing determines how the bit stream is translated in packets and how data is stored in the data buffers. It performs operations such as whitening and CRC handling. The framing is configured using `Set_PacketParams()` command.

4.5.1 LoRa modem

The LoRa modem uses spread spectrum modulation and forward error correction techniques to increase the range and reliability of sub-GHz radio communication. The LoRa modem provides improved co-channel rejection.

The LoRa modulation can be optimized for a given application by `Set_ModulationParams()` command, allowing a trade off between link budget, interference immunity, spectral occupation and nominal data rate.

The following parameters can be optimized:

- spreading factor (SF)
- modulation bandwidth (BW)
- error coding rate (CR)
- low data rate optimization (LDRO)

The LoRa symbol rate (Rs) is defined as $Rs = BW / 2^{SF}$

The transmitted signal is a constant envelope signal. Equivalently, one chip is sent per second per Hz of bandwidth.

a. The (G)FSK modem can be used for (G)MSK modulation in Tx and Rx when setting the frequency deviation to a quarter of the bit rate.

Spreading factor (SF)

The LoRa spread spectrum modulation is performed by representing each data bit of the packet payload by multiple chips of information. The rate at which the spread information is sent, is referred to as the symbol rate (Rs). The ratio between the nominal data rate and the chip rate is the spreading factor (SF). It represents the number of symbols per data bit.

The spreading factor must be known in advance on both transmit and receive side of the link.

The resulting signal to noise ratio (SNR) required at the receiver input, is influenced by the spreading factor. This allows the increase of the receiver sensitivity, and so increase the link budget and range.

A higher spreading factor provides a better receiver sensitivity, more link budget and longer range at the expense of a longer transmission time (see the table below).

Table 22. Spreading factor, chips/symbol and LoRa SNR

Spreading factor (SF)	5	6(1)	7(2)	8	9	10	11	12
2^{SF} (chips/symbol)	32	64	128	256	512	1024	2048	4096
LoRa demodulator SNR (dB)	–2.5	–5	–7	–9.5	–12	–14.5	–17	- 19

1. The SF6 is not backward compatible with earlier LoRa devices.
2. Default value.

For SF5 and SF6, due to the higher symbol rate, the minimum preamble length needed to ensure correct detection and demodulation of the received signal is 12 symbols.

Bandwidth (BW)

An increase in signal bandwidth allows the use of a higher effective data rate and reduces the transmission time at the expense of a reduced sensitivity, less link budget and shorter range. The LoRa modem operates at a programmable bandwidth (BW) around a programmable RF frequency (f_{rf}).

There are country dependent regulatory constraints on the permitted occupied bandwidth. The LoRa signal bandwidth refers to the double side bandwidth (DSB). The bandwidth selection range is given in the table below.

Table 23. LoRa bandwidth setting

LoRa BW setting	0	1	2	3	4	5	6	7	8(1)	9(1)
Bandwidth (kHz)	7.81	10.42	15.63	20.83	31.25	41.67	62.5	125	250	500

1. Bandwidth 250 kHz and 500 kHz are not available below a 400 MHz RF frequency.

Forward error correction coding rate (CR)

The sub-GHz radio communication reliability can be improved by performing forward error correction. This is particularly efficient in the presence of interference. The coding rate can be changed in response to channel condition. The coding rate information is included in the packet header for use by the receiver.

A higher coding rate provides better immunity to interference at the expense of longer transmission time. In normal conditions and factor of 4 / 5 provides the best trade off. In case of strong interference, a higher coding rate may be used.

The coding rate and overhead ratio is given in the table below.

Table 24. Coding rate and overhead ratio

CR setting	0(1)	1	2(2)	3	4
Coding rate (data bits/total coded bits)	4/4	4/5	4/6	4/7	4/8
Overhead ratio	1	1.25	1.5	1.75	2

1. No forward error correction.
2. Default value.

Low data rate optimization (LDRO)

For low data rates (typically high SF or low BW) and very long payloads (may last several seconds), the low data rate optimization (LDRO) can be enabled. This reduces the number of bits per symbol to the given SF minus 2, to allow the receiver to have a better tracking of the LoRa receive signal. Depending on the payload length, the low data rate optimization is usually recommended when the LoRa symbol time is equal or above 16.38 ms.

When using LoRa modulation, the total frequency drift over the packet time must be kept lower than Freq_drift_max:

$Freq\_drift\_max = BW / (3 \times 2^{SF})$

When possible, enabling the low data rate optimization (`Set_ModulationParams()` command), relaxes the total frequency drift over the packet time by 16:

$Freq\_drift\_optimise\_max = 16 \times Freq\_drift\_max$

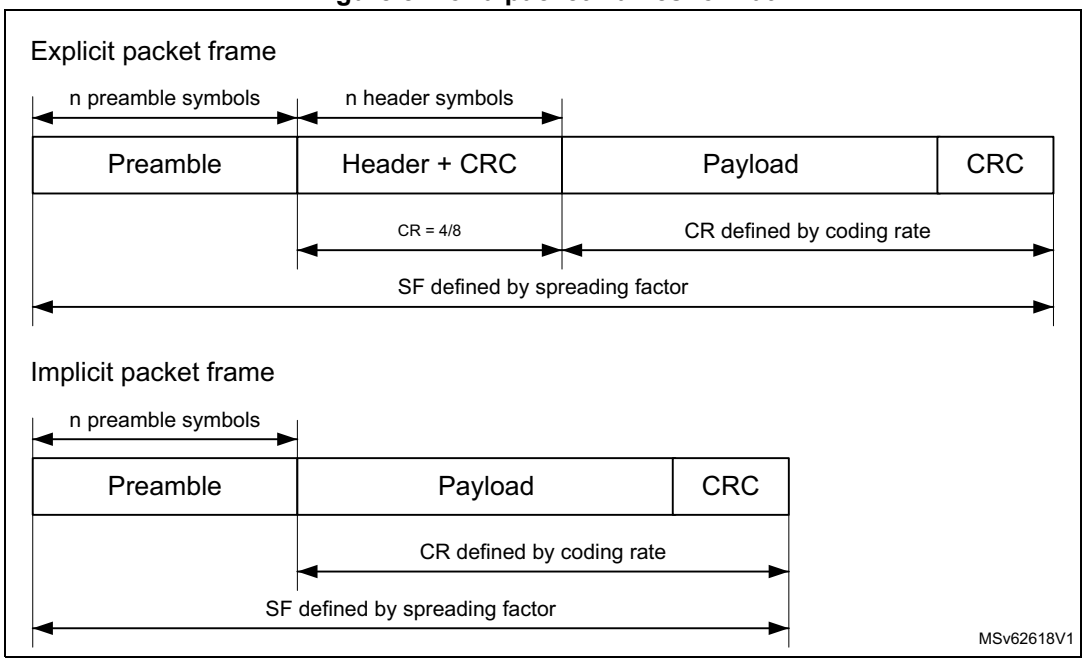
4.5.2 LoRa framing

The following types of LoRa packet formats are available:

- explicit header mode packet: includes a short header that contains information about the number of payload bytes, coding rate and presence of CRC.
- implicit header mode packet (no header)

The LoRa packet frames are illustrated in the figure below.

Figure 8. LoRa packet frames format



The LoRa packet frames start with a preamble that is used to synchronize the receiver with the received signal. By default, the packet is configured with a 12-symbol-long preamble sequence. The preamble length is programmable by `Set_PacketParams()` command. This allows the transmission of near arbitrarily long preamble sequences.

The receiver undertakes a preamble detection process that periodically restarts. For this reason, the preamble length at receiving side must be configured identical to the one at transmitting side. When the preamble length is not known at receiving side or when it varies, the maximum preamble length must be programmed at receiving side.

In the explicit packet format, the preamble is followed by a header with CRC, then followed by the payload and payload CRC.

In the implicit packet format, the preamble is directly followed by the payload and payload CRC.

The payload is a variable length field that contains the user data coded at the coding rate, either as specified in the explicit header or in the sub-GHz configuration when using implicit mode, by `Set_PacketParams()` and `Set_ModulationParams()` commands.

The payload can be followed by an optional payload CRC.

Explicit header mode

The default operation mode is the explicit header mode, where the header provides information on the payload. The header is transmitted using maximum forward error correction coding rate 4/8. It is protected by its own header CRC to allow the receiver to discard a packet up on receiving an invalid header.

Implicit header mode

In certain operation modes where the payload coding rate and CRC presence are fixed or known in advance, it can be advantageous to reduce transmission time by invoking implicit header mode. In this mode, the header is not present in the packet frame. The payload length, forward error correction coding rate and presence of the payload CRC must be configured on both sides of the sub-GHz radio link.

LoRa time-on-air

The total transmission time of packet can be calculated as follows:

$$\text{TotalTimeOnAir} = (\text{PreambleSymbols} + \text{NbSymbolPayload} + 4.25 + 8) \times (2^{SF} / \text{BW})$$

where PreambleSymbols is the number of preamble symbols programmed in `Set_PacketParams(PbLength)`.

The number of payload symbols can be calculated as follows:

$$\text{NbSymbolPayload} = \text{CEILING}(\text{NbSymbolPayloadFrac}, 4 + \text{CR})$$

where CEILING is the function that rounds up to the integer multiple of (4 + CR) immediately superior to the fractional first parameter.

The number of payload fractional symbols in Explicit mode can be calculated as follows:

$$\text{NbSymbolPayloadFrac} = ((\text{PL} \times 8 + \text{CRC} \times 16 - 4 \times (\text{SF} - 7)) \times (4 + \text{CR})) / (4 \times \text{SF})$$

The number of payload fractional symbols in Implicit mode can be calculated as follows:

$$\text{NbSymbolPayloadFrac} = ((\text{PL} \times 8 + \text{CRC} \times 16 - 4 \times (\text{SF} - 2)) \times (4 + \text{CR})) / (4 \times \text{SF})$$

where:

- CRC = 0 (no CRC) or 1 (16-bit CRC)
- CR = 0 to 4
- PL = 1 to 255, user data payload length in number of bytes
- SF = 7 to 12 spreading factor

LoRa data rates “raw data rate” and “real data rate” are defined as:

$$\text{RawBitRate} = ((\text{SF} \times \text{BW}) / 2^{SF}) \times (4 / (4 + \text{CR}))$$

$$\text{RealBitRate} = (\text{PayloadLength} \times 8) / \text{TotalTimeOnAir}$$

Channel activity detection (CAD)

The channel activity detection is used to detect the presence of a LoRa signal, by detecting a LoRa preamble or data symbols.

Once in channel activity detection mode, the band is scanned for a determined duration as set in `Set_CadParams()` command. If LoRa symbols are detected during this period, the channel activity detected IRQ is set.

The time needed to perform the channel activity detection depend on the LoRa modulation settings. For a given SF / BW, the typical CAD detection time can be selected to be 1, 2, 4, 8, or 16 symbols. The CAD duration time is further more extended with half a symbol.

4.5.3 FSK modem

The FSK modem provides a 2-FSK modulation over a range of data rates from 0.6 Kbit/s up to 300 Kbit/s. In receive mode, the bandwidth is automatically adjusted to match the selected data rate. In transmit mode, the frequency deviation is selected via the modulation

index. An optional Gaussian filter can be used. All modulation parameters are set using `Set_ModulationParams()` command.

The bit rate (or equivalent chip) is referenced to the HSE32 frequency and controlled by the BR parameter, defined as follows:

BR = 32 x $HSE32_{FREQ}$ / BitRate where $HSE32_{FREQ}$ = 32 MHz

FSK modulation is performed inside the RF-PLL bandwidth, by changing the fractional divider. The high resolution of the RF-PLL allows very narrow frequency deviation. The frequency deviation parameter Fdev, is one of the parameters in `Set_ModulationParams()`. Fdev is defined as follows:

Fdev= Fdev[Hz] / FreqStep where FreqStep = $HSE32_{FREQ}$ / 2^{25}

To ensure correct modulation, the following limit applies: (Fdev + BR/2) < BW.

The bandwidth must be chosen so that:

BW[DSB] ≥ BR + 2 x frequency deviation + frequency error

where frequency error = 2 x $HSE32_{FREQ}$ error

The FSK modem offers several pulse shape options in the PulseShape parameter.

4.5.4 MSK modem

The MSK modem provides a 2-MSK modulation over a range of data rates from 0.1 Kbit/s up to 10 Kbit/s. This MSK modulation is only available in transmit mode. The modulation index is automatically adjusted to 0.5. An optional Gaussian filter can be used. All modulation parameters are set using `Set_ModulationParams()` command.

The bit rate (or equivalent chip) is referenced to the HSE32 frequency and controlled by the BR parameter, defined as follows:

BR = $HSE32_{FREQ}$ / BitRate where $HSE32_{FREQ}$ = 32 MHz

4.5.5 Generic framing

The generic packet framing is used with the FSK and MSK modems.

The generic packet provides a conventional packet format for applications in proprietary NRZ coded, long range, low-energy communication links. The generic packet framing can be configured by `Set_PacketParams()` command and allows whitening based on pseudo random number generation, CRC operations and packet acknowledgment.

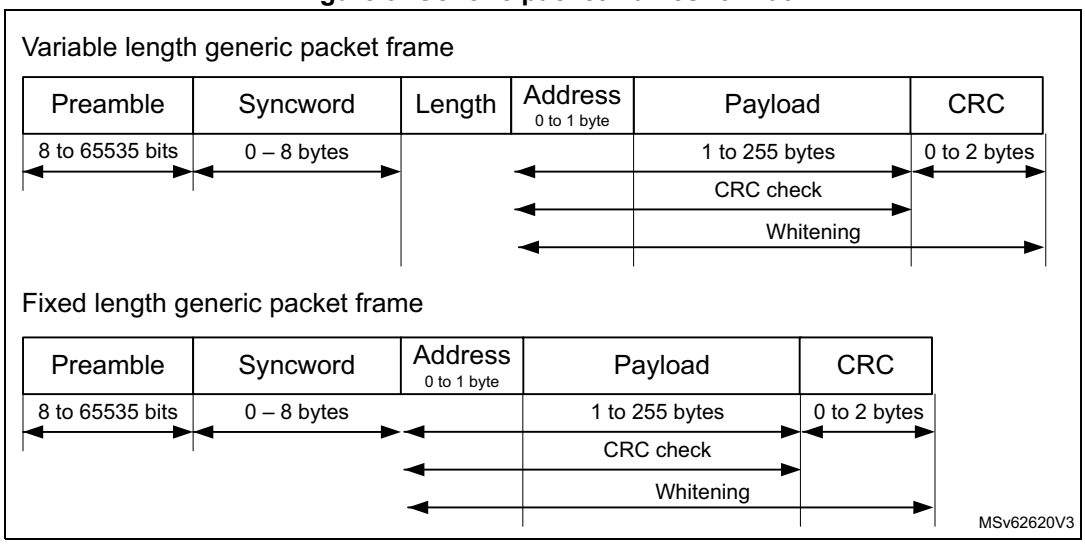
The following types of generic packet formats are available:

- Variable length packet: including a short header (named length) that contains information about the number of payload bytes
- Fixed length packet: no header

In both packet formats, the payload length is limited to 254 when the address filtering is activated.

The generic packet frames are illustrated in the figure below.

Figure 9. Generic packet frames format



1. The payload length can be extended beyond 255 bytes. Refer to *Section 4.6: Sub-GHz radio data buffer* for more details.

The generic packet frames start with a preamble that is used to synchronize the receiver with the received signal. The preamble length is programmable by `Set_PacketParams()` command.

The receiver undertakes a preamble detection process that periodically restarts. For this reason, the preamble length at the receiving side must be configured identical to the one of the transmitting side. When the preamble length is not known at receiving side or when it can vary, the maximum preamble length must be programmed at receiving side.

The preamble is followed by a syncword field with programmable length in `Set_PacketParams()` command.

The address is optional and can be used for a unicast when several devices share the same syncword.

In the variable length generic packet format, the syncword is followed by the length of the payload (If the Address field is present then Length must include this additional byte), followed by the payload and payload CRC.

In the fixed length generic packet format, the syncword is directly followed by the payload and payload CRC.

The payload is a variable length field that contains the user data either as specified in the variable length generic packet or in the sub-GHz radio configuration when using fixed length generic packet by `Set_PacketParams()`.

The payload can be followed by an optional payload CRC with programmable length in `Set_PacketParams()` command.

Endianness of the frame is MSB first.

Variable length generic packet mode

When the packet is of uncertain or variable length, the information on the payload length must be transmitted within the packet. For this, a header with the payload length information is transmitted after the syncword.

Fixed length generic packet mode

In certain operation modes where the payload length is fixed or known in advance, it may be advantageous to reduce transmission time by invoking fixed length generic packet mode. In this mode, the header is not present in the packet frame and the payload length must be configured on both sides of the sub-GHz radio link.

Node or broadcast address

The node or broadcast address are not considered part of the payload. They are added automatically when enabled by `Set_PacketParams()` command. Adding these fields allows the user to perform additional packet filtering at the header level.

Whitening

Whitening is based on a 9-bit LFSR and used to whiten the payload and, when present, the header and CRC. Whitening limits a sequence of consecutive 1 or 0 bit to nine. The whitening polynomial is $x^9 + x^5 + 1$, and can be initialized with the whitening initial value in WHITEINI[8:0].

CRC

CRC computation can be configured for polynomial and initial value, and allows inversion. Configuration is done through `Set_PacketParams()` command.

The flexible CRC configuration allows any standard CRC or proprietary CRC to be generated and checked, for example:

- IBM CRC configuration
  - CRC polynomial 0x8005
  - CRC initial value 0xFFFF
  - 2-byte length CRC
- CCIT CRC configuration
  - CRC polynomial 0x1021
  - CRC initial value 0x1D0F
  - 2-byte length inverted CRC

4.5.6 BPSK modem

The BPSK modem provides a BPSK modulation for data rates of 100 and 600 bit/s. BPSK modulation is only available in transmit mode. The modulation offers a raise-cosine pulse shape filter with a BT of 0.5. The BPSK modulator can be changed into DBPSK modulation by some host pre-processing on the data to be transmitted.

All modulation parameters are set by `Set_ModulationParams()` command.

The bit rate (or equivalent chip) is controlled by Br parameter, defined as follows:

Br = $HSE32_{FREQ}$ / BitRate where $HSE32_{FREQ}$ = 32 MHz.

4.5.7 BPSK framing

The BPSK packet framing is used with the BPSK modem. The BPSK packet framing can be configured by `Set_PacketParams()` command and allows the total frame length definition. The full packet (preamble, synch word, device id to CRC) must be provided in the transmit data buffer.

4.6 Sub-GHz radio data buffer

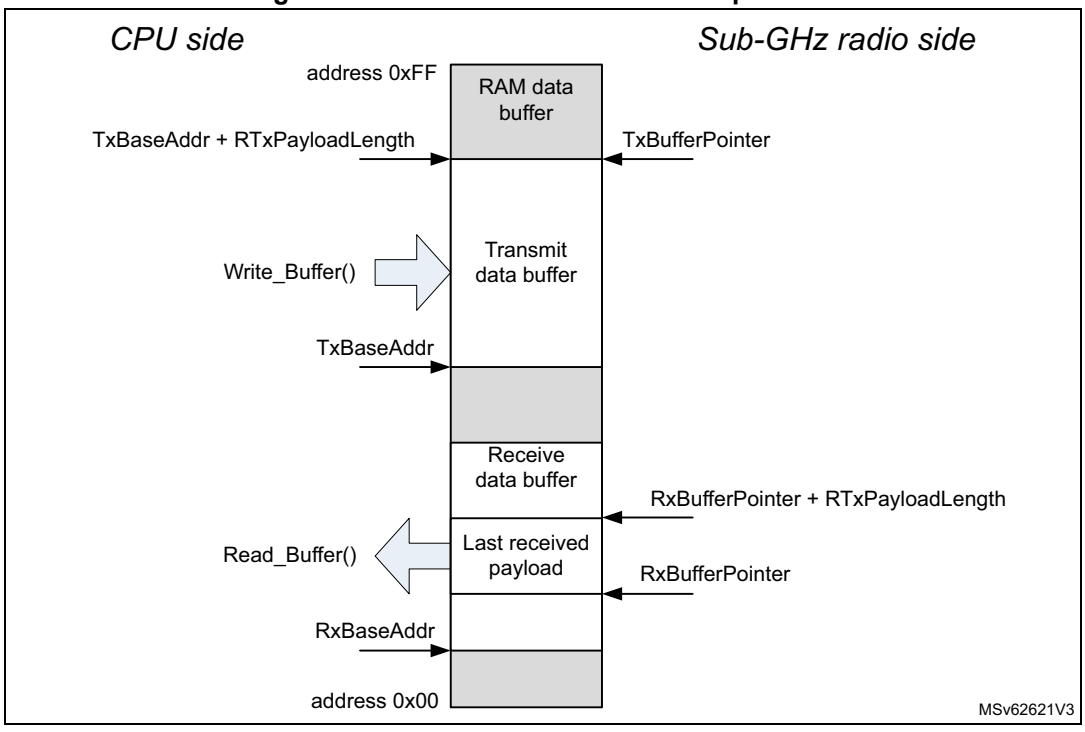
The sub-GHz radio uses a 256-byte RAM data buffer that is accessible by the CPU through the SUBGHZSPI interface in all sub-GHz radio operating modes except Sleep and Deep-Sleep. The RAM data buffer is cleared in Deep-Sleep mode, optionally retained in Sleep mode and always retained in all other modes. In Sleep and Deep-Sleep modes, the TxBaseAddr and RxBaseAddr values are lost. In order to retrieve data after Sleep mode retention, the default values must be used (TxBaseAddr = 0x80 and RxBaseAddr = 0x00), or RxPayloadLength and RxBufferPointer must be stored in the CPU memory.

On die revision Y and later, the PayloadLength can be updated during transmission. This capability allows the support of long packets.

On previous die revisions, the PayloadLength is fixed (latched in the radio) after a `SetTx()` command.

The RAM data buffers are fully configurable. They allow access to the transmit data buffer and the last receive data buffer. The RAM data buffer organization is depicted in the figure below.

Figure 10. Sub-GHz RAM data buffer operation



The RAM data buffer is accessed by `Write_Buffer()` and `Read_Buffer()` commands. The offset parameter defines the address in the SRAM data buffer. To read the first byte

from the receive data buffer, the offset must be set to the RxBufferPointer value. To write to the first byte in the transmit data buffer, the offset must be set to the TxBaseAddr value.

The RAM data buffer has a circular nature: any address increment exceeding 0xFF wraps around to address 0x00.

4.6.1 Receive data buffer operation

In receive mode, RxBaseAddr, configured through `Set_BufferBaseAddress()`, determines the receive buffer offset in the sub-GHz radio RAM. RxBaseAddr can be set anywhere within the RAM data buffer. If needed, the whole 256-byte RAM data buffer can be used.

For the first received packet, RxBufferPointer is set to RxBaseAddr. The first byte of the first received packet payload data is written at the RxBufferPointer address. The last byte of the received packet payload data is written at the address determined by RxBufferPointer + RxPayloadLength. RxBufferPointer and RxPayloadLength can be read by `Get_RxBufferStatus()` command.

In single receive and listening modes, RxBufferPointer is automatically initialized to RxBaseAddr for each new reception when the sub-GHz radio enters RX mode.

In continuous receive mode, RxBufferPointer is continuously incremented from the position at which it was left by the previous received packet. In this mode, subsequent received packets are stored continuous in the RAM data buffer.

Caution: If the amount of received data exceeds the defined receive buffer size, other data in the RAM are overwritten.

Receive data are written to the receive data buffer regardless of the received CRC status. When the received CRC fails, post processing on the incorrectly received data can still be performed by the CPU.

Caution: When the received packet length exceeds the area allocated to the receive data buffer, it may overwrite the transmit data buffer area in the RAM data buffer.

4.6.2 Transmit data buffer operation

In transmit mode, TxBaseAddr determines the transmit buffer offset in the sub-GHz radio RAM. For each transmission, when the sub-GHz radio enters TX mode, TxBufferPointer is automatically initialized to TxBaseAddr. Each time a payload byte is transmitted, TxBufferPointer is incremented until all bytes, according the PayloadLength set by the `Set_PacketParams()` command, are transmitted.

4.7 Sub-GHz radio operating modes

After a sub-GHz radio reset, a startup phase is initiated. As BUSY is active during this phase, commands cannot be accepted. When supply and clock are available to the sub-GHz radio, BUSY is deactivated and the CPU can take control.

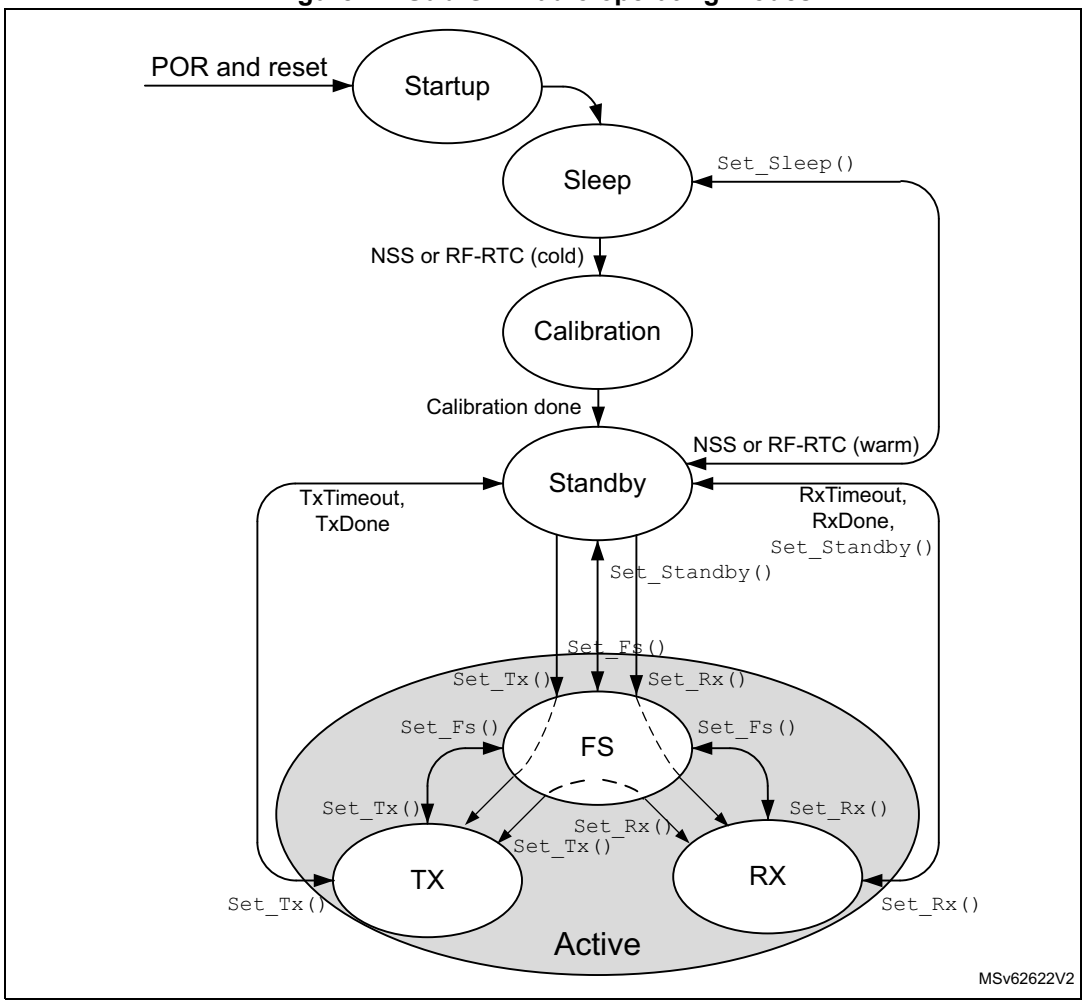
The sub-GHz radio provides the following operating modes:

- **Sleep mode**
  - **Deep-Sleep**: all sub-GHz radio clocks are turned off and data memory is lost
  - RC 13 MHz clock turned off

  - RC 64 kHz and timers can be kept running (optional)
  - Optional registers and data memory retained
- **Calibration mode**
  - intermediate mode between Deep-Sleep or Sleep, and Standby
  - used to calibrate the sub-GHz radio RC 64 kHz, sub-GHz radio RC 13 MHz, RF-PLL, RF-ADC and image
- **Standby mode**
  - sub-GHz radio clocked by its internal RC 13 MHz clock
  - HSE32 clock can be kept running (optional)
  - data memory retained
- **Active mode (FS, TX, RX)**
  - Frequency synthesis (FS) mode: RF-PLL switched on
  - Transmit (TX) mode: the power amplifier (PA) ramped and data transmitted from the data buffer according the selected modulation scheme
  - Receive (RX) mode: sub-GHz radio looking for incoming data packets

The sub-GHz radio operating modes are shown in the figure below.

Figure 11. Sub-GHz radio operating modes



4.7.1 Startup mode

At POR or after a sub-GHz radio reset, the Startup mode is entered. BUSY is set. When internal supply and clocks become available, the sub-GHz radio enters Sleep mode.

4.7.2 Sleep mode

In Sleep mode, only the sub-GHz radio startup and Sleep control is operational and the configuration is lost. BUSY is set. Optionally, configuration registers and memories may be kept in retention. The RC 64 kHz and sub-GHz RTC may be kept running.

The Sleep mode provides the following sub-mode and options:

- Deep-Sleep mode: all sub-GHz radio function off, controlled by `set_Sleep()` command
- Sleep with the sub-GHz radio RC 64 kHz kept on: configuration registers can be retained as configured by `Set_Sleep()` command
- Sleep with the data RAM retained, controlled by `Set_Sleep()` command

Following a POR sub-GHz radio reset, the Deep-Sleep mode is entered from Startup mode. Sleep mode can be entered from Standby mode by `Set_Sleep()` command.

Caution: After `Set_Sleep()` command, the sub-GHz radio cannot receive any SPI commands. The user must guarantee that the sub-GHz radio SPI NSS is not set low during 500 µs.

Exit Sleep mode can be done:

- on a firmware request via the sub-GHz radio SPI NSS signal (keeping sub-GHz radio SPI NSS low for at least 20 µs)
- on a request from the sub-GHz radio RTC timer generating an end-of-count event (corresponding to duty cycled operation)

When the sub-GHz radio configuration registers are retained, a warm start is performed when exiting Sleep mode. During a warm start, the configuration registers are restored with their retained value an the Calibration state is skipped.

4.7.3 Calibration mode

On a cold start, when transitioning from Deep-Sleep or Sleep mode to Standby mode, the intermediate Calibration mode is entered. In Calibration mode, BUSY is set to indicate that the sub-GHz radio is busy and cannot accept any SPI command.

The calibration phase consists of the following operations:

- sub-GHz radio RC 64 kHz frequency calibration
- sub-GHz radio RC 13 MHz frequency calibration
- RF-PLL modulation path calibration
- RF-ADC calibration
- image calibration

The total calibration time is 1.6 ms. All calibration results are stored in data RAM.

When the data RAM is retained in Sleep mode, the sub-GHz radio retrieves the calibration data and then transitions to Standby mode without repeating the calibration phase.

Once the calibration is finished, BUSY is deactivated and the sub-GHz radio enters Standby with RC 13 MHz mode.

When in Standby mode, the calibration of different blocks can be requested by `Calibrate()` command.

Image calibration for specific frequency bands

The image calibration is performed as part of the calibration process, by default in the band 902 - 928 MHz. The image calibration can furthermore be requested in any band by `CalibrateImage()` command.

Image calibration must be redone when the RF frequency is changed significantly and when the temperature changes more than 20 °C.

4.7.4 Standby mode

The Standby mode is entered from the Calibration mode and can furthermore be entered by `Set_Standby()` command.

In Standby mode, BUSY is cleared. The software must configure the sub-GHz radio for FS, TX, RX or Sleep mode. By default, in Standby mode, the sub-GHz radio is clocked by the sub-GHz RC 13 MHz. For time critical applications, HSE32 can be turned on. The use of RC 13 MHz or HSE32 is selected by `Set_Standby()` command.

If the SMPS is used during the sub-GHz transmission or reception, SMPS must be enabled in the Standby with RC 13 MHz mode. The SMPS is selected by `Set_RegulatorMode()` command. When selected, the SMPS is enabled automatically when entering the Standby with HSE32 mode. If the SMPS has been enabled by the CPU with PWR_CR5.SMPSEN, the SMPS remains enabled in all sub-GHz radio operating modes including transmission and reception.

4.7.5 Frequency synthesis mode (FS)

When entering FS mode, BUSY is set. In FS mode, the RF-PLL is activated. Once RF-PLL is locked, BUSY goes low. The sub-GHz radio may be requested to enter this mode by `Set_Fs()`.

Due to the low-IF architecture, the transmit and receive RF-PLL frequencies are different. The receive RF-PLL frequency is equal to the transmit frequency minus the intermediate frequency offset.

4.7.6 Transmit mode (TX)

The TX mode can be requested to be entered from Standby mode. In this case, the sub-GHz radio enters first FS mode to lock the RF-PLL, and subsequently TX mode.

In TX mode after ramping-up the power amplifier (PA), data from the data buffer is transmitted. The sub-GHz radio operates in one of the following transmit modes:

- single transmit mode
- single transmit with timeout mode: timeout used as safety, when the transmit is aborted or does not succeed. The timeout informs the software of such events.

The sub-GHz radio returns automatically to Standby mode after the last data bit of the packet is transmitted or after the timeout expires.

TX mode entry is requested by `Set_Tx()` command.

When entering TX mode, BUSY is set. In TX mode, BUSY is cleared when the PA ramped up and preamble transmission starts.

PA ramping

The PA ramping time can be selected while setting the output power, by `Set_TxParams()`.

4.7.7 Receive mode (RX)

The RX mode can be requested to be entered from Standby mode. In this case, the sub-GHz radio enters first FS mode to lock the RF-PLL, and subsequently RX mode.

In RX mode, LNA, RF-PLL and selected modem are enabled. The receive mode provides the following operating modes:

- Continuous mode: The sub-GHz radio remains in RX mode and waits for incoming packets reception until the software terminates RX mode.
- Single mode: The sub-GHz radio remains in RX mode until a packet reception and enters automatically Standby mode after.
- Single with timeout mode: The sub-GHz radio remains in RX mode until a packet reception or until the timeout expires, and enters automatically Standby mode after.
- Listening mode: The sub-GHz radio repeatedly switches between RX single with timeout mode and Sleep mode, until an IRQ is triggered.

RX mode entry is requested by `Set_Rx()` command.

When entering RX mode, BUSY is set. In RX mode, BUSY is cleared when the receiver is ready to receive data.

The receiver provides a power-saving mode at the expense of reduced sensitivity. The receiver power-saving mode is controlled by the SUBGHZ_RXGAINR register.

4.7.8 Active mode switching time

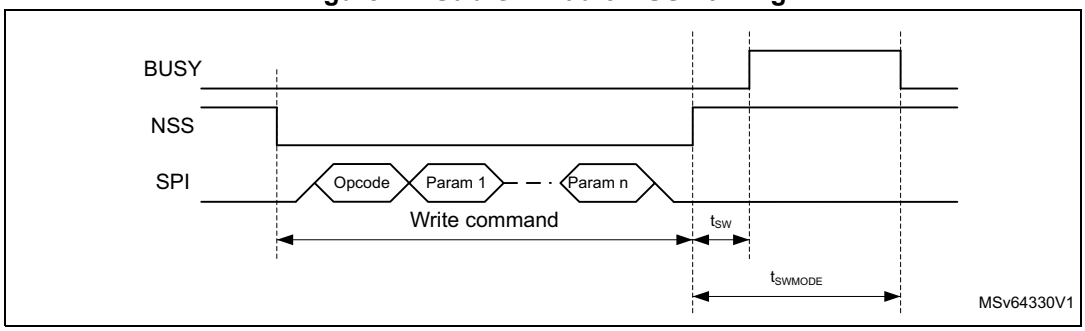
At each transaction with the sub-GHz radio (register read/write operation or mode switching), BUSY is set during the transaction and while the sub-GHz radio is processing the command. BUSY is cleared once the command processing is finished or reached a stable operating mode, and the sub-GHz radio is ready to receive a new command.

All write commands activate BUSY. For read commands, BUSY remains low.

Switching time (t_{SWMODE}) is defined as the time to process the command or reach a stable operating mode, starting from the sub-GHz radio SPI NSS rising edge, ending the SPI command transaction, until BUSY goes inactive. There is a small delay (t_{SW}) between the sub-GHz radio SPI NSS rising edge, ending the SPI command transaction, and when BUSY is set. The maximum time for t_{SW} is 600 ns.

BUSY timing is shown in the figure below.

Figure 12. Sub-GHz radio BUSY timing



For the different mode transitions, typical busy timing values are given in the table below.

Table 25. Operation mode transition BUSY switching time

Mode transition	SPI command (sub-GHz radio event)	t_{SWMODE} typical (µs)
Sleep-to-Standby (no data retention)	SPI NSS low 20 µs	3500
Sleep-to-Standby (with data retention)	SPI NSS low 20 µs (RTC end-of-count)	340
Standby-to-Standby with HSE32	`Set_Standby()`	31
Standby (HSE32 off)-to-FS(1)	`Set_Fs()`	50
Standby (HSE32 off)-to-TX(2)	all `Set_Tx()`	126
Standby (HSE32 off)-to-RX(3)	`Set_Rx()`, `Set_Cad()`	83
Standby (HSE32 on)-to-FS(1)	`Set_Fs()`	40
Standby (HSE32 on)-to-TX(2)	all `Set_Tx()`	105
Standby (HSE32 on)-to-RX(3)	`Set_Rx()`, `Set_Cad()`	62
FS-to-TX(2)	all `Set_Tx()`	76
FS-to-RX(3)	`Set_Rx()`, `Set_Cad()`	41

1. When entering FS mode, BUSY is cleared to 0 when RF-PLL is locked.
2. When entering TX mode, BUSY is cleared to 0 when the PA ramps up and the transmission of preamble starts.
3. When entering RX mode, BUSY is cleared to 0 when the receiver is ready to receive data.

4.8 Sub-GHz radio SPI interface

The sub-GHz radio SPI slave interface gives access to the sub-GHz radio configuration, registers and buffer memory, through SPI commands. It is connected to the SUBGHZSPI master interface peripheral on the CPU bus matrix.

For a write access, an opcode byte is sent followed by sending the command parameter bytes.

For a read access, an opcode byte is sent followed by receiving data bytes.

For each access, the sub-GHz radio SPI NSS goes low at the start of the transfer and is set high at the end, after all bytes have been transfered.

The following transaction types are supported:

- configuration transaction: provides the CPU with a direct access to control registers. Used to write or read sub-GHz radio configuration registers or buffer memory.
- command transaction: requires more complex, non-atomic operations such as packet transmission and reception or operating mode changes

BUSY is used to indicate the status of the sub-GHz radio and its ability (or not) to receive a SPI transaction. Prior to issuing a new SPI transaction, the CPU must check the BUSY status to make sure a new transaction can be received by the sub-GHz radio.

4.8.1 Sub-GHz radio command structure

The SPI command structure consists of an opcode and parameters when writing data to the sub-GHz radio, and consists of a status when reading data.

In case of a write command that does not require any parameters, the CPU sent only an opcode over the sub-GHz radio SPI interface.

In case of a write command that requires parameters, the opcode is followed immediately by the parameter bytes. All parameters must be sent before the sub-GHz radio SPI NSS rising edge, that terminates the command.

In case of a read command, the opcode is followed immediately by the status bytes. All status must be received before the sub-GHz radio SPI NSS rising edge, that terminates the command.

Command structure values are given in the table below.

Table 26. Command structure

Command type	Byte 0	Byte 1:n
Write command without parameter	Opcode	Not applicable
Write command with parameters	Opcode	Parameters
Read command	Opcode	data

4.8.2 Register and buffer access commands

Write_Register() command

`Write_Register(Addr, Data0, Data1, to Datan)` allows a block of bytes to be written in a contiguous memory area, starting from the specified address. The address is auto incremented after each byte.

0	1	2	3	...	n+3
Opcode	Addr[15:0]		Data0[7:0]	...	Datan[7:0]
w	w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x0D

bytes 2:1 bits 15:0 **Addr[15:0]**: first write address

byte 3 bits 7:0 **Data0[7:0]**: data to write to first address

... ...

byte n+3 bits 7:0 **Datan[7:0]**: data to write to address + n (n = number of bytes to write)

Read_Register() command

`Read_Register(Addr, Status, Data0, Data1, to Datan)` allows a block of bytes to be read in a contiguous memory area starting from the specified address. The address is auto incremented after each byte.

0	1	2	3	4	...	n+4
Opcode	Addr]15:0]		Status[7:0]	Data0[7:0]	...	Datan[7:0]
w	w	w	r	r	r	r

byte 0 bits 7:0 **Opcode**: 0x1D

bytes 2:1 bits 15:0 **Addr[15:0]**: first read address

byte 3 bits 7:0 **Status[7:0]**: see *Get_Status() command*

byte 4 bits 7:0 **Data0[7:0]**: data read from first address

... ...

byte n+4 bits 7:0 **Datan[7:0]**: data read from address + n (n = number of bytes to read)

Write_Buffer() command

`Write_Buffer(Offset, Data0, Data1, to Datan)` allows transmit packet payload data to be written to a contiguous data memory area, starting from the specified offset. The offset is auto incremented after each byte. When the offset exceeds the value 255, it is wrapped around to 0 (providing a 256-byte circular buffer).

0	1	2	...	n+2
Opcode	Offset[7:0]	Data0[7:0]	...	Datan[7:0]
w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x0E

byte 1 bits 7:0 **Offset[7:0]**: first write address offset

byte 2 bits 7:0 **Data0[7:0]**: data to write to offset address

... ...

byte n+2 bits 7:0 **Datan[7:0]**: data to write to offset address + n (n = number of bytes to write)

Read_Buffer() command

`Read_Buffer(Offset, Status, Data0, Data1, to Datan)` allows receive packet payload data to be read from a contiguous data memory area, starting from the specified

offset. The offset is auto incremented after each byte. When the offset exceeds the value 255, it is wrapped around to 0 (providing a 256 byte circular buffer).

0	1	2	3	...	n+3
Opcode	Offset[7:0]	Status[7:0]	Data0[7:0]	...	Datan[7:0]
w	w	r	r	r	r

byte 0 bits 7:0 **Opcode**: 0x1E

byte 1 bits 7:0 **Offset[7:0]**: first read address offset

byte 2 bits 7:0 **Status[7:0]**: see *Get_Status() command*

byte 3 bits 7:0 **Data0[7:0]**: data read from offset address

... ...

byte n+3 bits 7:0 **Datan[7:0]**: data read from offset address + n (n = number of bytes to read)

4.8.3 Operating mode commands

Set_Sleep() command

`Set_Sleep(SleepCfg)` is used to set the sub-GHz radio in Sleep mode. This command is only accepted in Standby mode. The SleepCfg parameter allows some optional functions to be maintained in Sleep mode.

0	1
Opcode	SleepCfg
w	w

byte 0 bits 7:0 **Opcode**: 0x84

byte 1 bits 7:3 Reserved, must be kept at reset value.

bit 2 **SleepCfg_Start**: Sub-GHz radio startup selection

0: cold startup when exiting Sleep mode, configuration registers reset

1: warm startup when exiting Sleep mode, configuration registers kept in retention

Note: Only the configuration of the activated modem, before going to Sleep mode, is retained. The configuration of the other modes is lost and must be re-configured when exiting Sleep mode.

bit 1 Reserved, must be kept at reset value.

bit 0 **SleepCfg_RTCEn**: Sub-GHz radio RTC wakeup enable

0: Sub-GHz radio RTC wakeup disabled

1: Sub-GHz radio RTC wakeup enabled

Set_Standby() command

`Set_Standby(StandbyCfg)` is used to set the sub-GHz radio in Standby mode. The StandbyCfg parameter allows some optional functions to be selected in Standby mode.

0	1
Opcode	StandbyCfg
w	w

byte 0 bits 7:0 **Opcode**: 0x80

byte 1 bits 7:1 Reserved, must be kept at reset value.

bit 0 **StandbyCfg_StandbyClk**: set clock in Standby mode
0: RC 13 MHz used in Standby mode
1: HSE32 used in Standby mode (Standby with HSE32)

Set_Fs() command

`Set_Fs()` is used to set the sub-GHz radio in FS mode. This command allows the RF-PLL test.

0
Opcode
w

byte 0 bits 7:0 **Opcode**: 0xC1

The RF-PLL frequency must be set by `Set_RfFrequency()` command prior sending `Set_Fs()`.

Set_Tx() command

`Set_Tx(Timeout)` is used to set the sub-GHz radio in TX mode.

0	1	2	3
Opcode	Timeout[23:0]		
w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x83

bytes 3:1 bits 23:0 **Timeout[23:0]**: Transmit packet timeout
0x000000: Timeout disabled
0x000001 - 0xFFFFFF: Timeout enabled, resolution 15.625 µs

Time-out duration is computed as follows:

Time-out duration = Timeout x 15.625 µs (maximum time-out duration = 262.14 s)

When `Set_Tx(Timeout)` is sent in Standby or Receive mode, the sub-GHz radio passes through the FS mode (no need to send `Set_Fs()`). In this case, the RF-PLL frequency must be set by `Set_RfFrequency()` prior sending `Set_Tx(Timeout)`.

Set_Rx() command

`Set_Rx(Timeout)` is used to set the sub-GHz radio in Receive mode.

0	1	2	3
Opcode	Timeout[23:0]		
w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x82

bytes 3:1 bits 23:0 **Timeout[23:0]**: Transmit packet timeout
0x000000: timeout disabled
0x000001 - 0xFFFFFE: timeout enabled, single packet receive mode, resolution 15.625 µs
0xFFFFFF: timeout disabled, continuous receive mode

Time-out duration is computed by the following formula:

Time-out duration = Timeout x 15.625 µs (maximum Time-out duration = 262.14 s)

When `Set_Rx(Timeout)` is sent in Standby mode or Transmit mode, the sub-GHz radio passes through the FS mode (no need to send `Set_Fs()`. In this case, the RF-PLL frequency must be set by `Set_RfFrequency()` prior sending `Set_Rx(Timeout)`.

Set_StopRxTimerOnPreamble() command

`Set_StopRxTimerOnPreamble(RxTimeoutStop)` allows the selection of the receiver event on which the receiver timeout timer (in `Set_Rx()`), is stopped.

0	1
Opcode	RxTimeoutStop
w	w

byte 0 bits 7:0 **Opcode**: 0x9F

byte 1 bits 7:1 Reserved, must be kept at reset value.

bit 0 **RxTimeoutStop**: receiver timeout timer stop event selection
0: receive timeout stopped on synchronization word detection in generic packet mode or header detection in LoRa packet mode
1: receive timeout stopped on preamble detection

Caution: When the receiver timeout is selected to be stopped on preamble detection, the sub-GHz radio remains in Receive mode until a packet is received. A false preamble detection may cause the sub-GHz radio to remain in Receive mode for an unexpected long period, until stopped by a mode configuration command.

Set_RxDutyCycle() command

`Set_RxDutyCycle(RxPeriod, SleepPeriod)` is used to set the sub-GHz radio receiver in listening mode, regularly looking for new packets. This command must be sent in Standby mode. This command is only functional with FSK and LoRa packet type.

The following steps are performed:

1. Save sub-GHz radio configuration.
2. Enter Receive mode and listen for a preamble for the specified RxPeriod period.
3. Upon the detection of a preamble, the RxPeriod timeout is stopped and restarted with the value 2 x RxPeriod +SleepPeriod. During this new period, the sub-GHz radio looks for the detection of a synchronization word when in (G)FSK modulation mode, or a header when in LoRa modulation mode.
4. If no packet is received during the listen period defined by 2 x RxPeriod + SleepPeriod, the Sleep mode is entered for a duration of SleepPeriod. At the end of the receive period, the sub-GHz radio takes some time to save the context before starting the sleep period.
5. After the sleep period, a new listening period is automatically started. The sub-GHz radio restores the sub-GHz radio configuration and continuous with step 2.

The listening mode is terminated in one of the following cases:

- if a packet is received during the listening period: the sub-GHz radio issues a RxDone interrupt and enters Standby mode.
- if `SetStandby()` is sent during the listening period or after the sub-GHz has been requested to exit Sleep mode by sub-GHz radio SPI NSS

0	1	2	3	4	5	6
Opcode	RxPeriod[23:0]			SleepPeriod[23:0]		
w	w	w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x94

bytes 3:1 bits 23:0 **RxPeriod[23:0]**: Receive duration
0x000000: Receiver duration disabled, receiver remaining active until a packet is detected
0x000001 - 0xFFFFFF: Receive duration, resolution 15.625 µs

bytes 6:4 bits 23:0 **SleepPeriod[23:0]**: Sleep duration, resolution 15.625 µ

Receive period duration is computed as follows:

Receive period duration = RxPeriod x 15.625 µs (max receiver duration = 262.14 s)

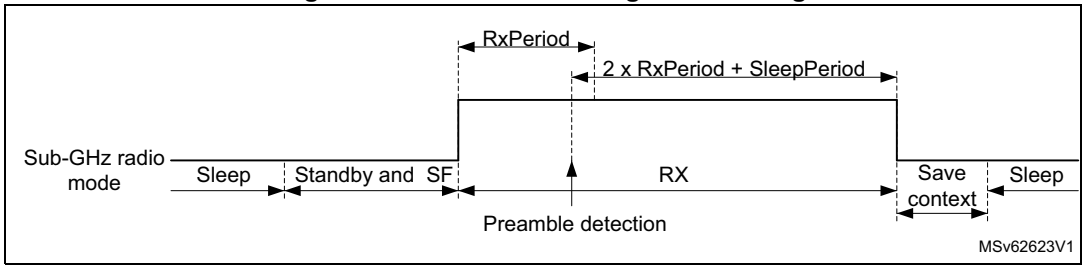
Sleep period duration is computed by the following formula:

Sleep period duration = SleepPeriod x 15.625 us (max sleep duration = 262.14 s)

For correct operation, the following must be respected:

- (G)FSK modulation: $T_{preamble} + T_{syncword} \leq 2 \times RxPeriod + SleepPeriod$
- LoRa modulation: $T_{preamble} + T_{header} \leq 2 \times RxPeriod + SleepPeriod$

Figure 13. Receiver listening mode timing



Set_Cad() command

`Set_Cad()` is used to detect the channel activity and can only be used with LoRa packet types. The channel activity detection (CAD) is a specific LoRa operation mode, where the sub-GHz radio searches for a LoRa radio signal. After the search is completed, the Standby mode is automatically entered, CAD is done and IRQ is generated. When a LoRa radio signal is detected, the CAD detected IRQ is also generated.

0
Opcode
w

byte 0 bits 7:0 **Opcode**: 0xC5

The length of the search must be configured with `Set_CadParams()` prior sending `Set_Cad()`.

Set_TxContinuousWave() command

`Set_TxContinuousWave()` is a test command to generate a continuous transmit tone at the RF-PLL frequency. The sub-GHz radio remains in continuous transmit tone mode until a mode configuration command is received.

0
Opcode
w

byte 0 bits 7:0 **Opcode**: 0xD1

Set_TxContinuousPreamble() command

`Set_TxContinuousPreamble()` is a test command to generate an infinite preamble at the RF-PLL frequency.

The preamble is an alternating 0s and 1s sequence in generic (G)FSK and (G)MSK modulations. The preamble is symbol 0 in LoRa modulation.

The sub-GHz radio remains in infinite preamble mode until a mode configuration command is received.

0
Opcode
w

byte 0 bits 7:0 **Opcode**: 0xD2

4.8.4 Sub-GHz radio configuration commands

Set_PacketType() command

`Set_PacketType(PktType)` allows the selection of packet frame format. This command must be the first command of a sub-GHz radio configuration sequence.

Changing from one sub-GHz radio configuration to another is done using `Set_PacketType()`. The parameters from the previous sub-GHz radio configuration are lost. The switch from one configuration mode to another is only accepted in Standby mode.

0	1
Opcode	PktType
w	w

byte 0 bits 7:0 **Opcode**: 0x8A.

byte 1 bits 7:2 Reserved, must be kept at reset value.

bits 1:0 **PktType[1:0]**: Packet type definition
- 0: FSK generic packet type
- 1: LoRa packet type
- 2: BPSK packet type
- 3: MSK generic packet type
- Other: reserved

Get_PacketType() command

`Get_PacketType(Status, PktType)` returns information on the selected packet frame format.

Changing from one sub-GHz radio configuration to another is done using `Set_PacketType()`. The parameters from the previous sub-GHz radio configuration are lost. The switch from one configuration mode to another is only accepted in Standby mode.

0	1	2
Opcode	Status[7:0]	PktType
w	r	r

byte 0 bits 7:0 **Opcode**: 0x11

byte 1 bits 7:0 **Status[7:0]**: see *Get_Status() command*

byte 2 bits 7:2 Reserved, must be kept at reset value.

bits 1:0 **PktType[1:0]**: Packet type definition
- 0: FSK generic packet type
- 1: LoRa packet type
- 2: BPSK packet type
- 3: MSK generic packet type
- Others: reserved

Set_RfFrequency() command

`Set_RfFrequency(RfFreq)` is used to lock the RF-PLL frequency to the transmit and receive frequency.

0	1	2	3	4
Opcode	RfFreq[31:0]			
w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x86

bytes 4:1 bits 31:0 **RfFreq[31:0]**: RF frequency

RF-PLL frequency = $32e^6$ x RFfreq / 2^{25}

Set_TxParams() command

`Set_TxParams(Power, RampTime)` is used to set the transmit output power and the PA ramp-up time.

0	1	2
Opcode	Power[7:0]	RampTime[7:0]
w	w	w

byte 0 bits 7:0 **Opcode**: 0x8E

byte 1 bits 7:0 **Power[7:0]**: Output power setting

LP PA selected in `Set_PaConfig()`
0x0E:+ 14 dB
...
0x00: 0 dB
...
0xEF: - 17 dB
Others: reserved
HP PA selected in `Set_PaConfig()`
0x16:+ 22 dB
...
0x00: 0 dB
...
0xF7: - 9 dB
Others: reserved

byte 2 bits 7:0 **RampTime[7:0]**: PA ramp time for FSK, MSK and LoRa modulation

0x00: 10 µs
0x01: 20 µs
0x02; 40 µs
0x03: 80 µs
0x04: 200 µs
0x05: 800 µs
0x06: 1700 µs
0x07: 3400 µs
Others: reserved

Note: In BPSK mode, the ramping time is specific and RampTime[7:0] is not used.

Set_PaConfig() command

`Set_PaConfig(PaDutyCycle, HpMax, PaSel, 0x01)` is used to customize the maximum output power and PA efficiency.

0	1	2	3	4
Opcode	PaDutyCycle[2:0]	HpMax[2:0]	PaSel	0x01
w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x95

byte 1 bits 7:3 Reserved, must be kept at reset value.

bits 2:0 **PaDutyCycle[2:0]**: PA duty cycle (conduit angle) control

Duty cycle = 0.2 + 0.04 x PaDutyCycle[2:0] (see *Table 27* for settings)

Caution: The following restrictions must be observed to avoid over-stress on the PA:
- LP PA mode with synthesis frequency >400 MHz, PaDutyCycle must be ≤ 0x7.
- LP PA mode with synthesis frequency < 400 MHz, PaDutyCycle must be ≤ 0x4.
- HP PA mode, PaDutyCycle must be ≤ 0x4.

byte 2 bits 2:0 **HpMax[2:0]**: HP PA output power (see *Table 27* for settings)

bits 7:3 Reserved, must be kept at reset value.

byte 3 bits 7:1 Reserved, must be kept at reset value.

bit 0 **PaSel**: PA selection.
0: HP PA selected
1: LP PA selected (default)

byte 4 bits 7:0 0x01

PA optimal settings given in the table below must be used to maximize the PA efficiency for the maximum targeted output power. Matching network determination must be done using these settings (see the application note AN5457 for more details).

Table 27. PA optimal setting and operating modes

Output power (dBm)	PA mode	Set_PaConfig()			Set_TxParams()
		PaDutyCycle [2:0]	HpMax[2:0]	PaSel	Power
+ 15	LP	0x6	0x0	1	0x0E
+ 14		0x4	0x0	1	0x0E
+ 10		0x1	0x0	1	0x0D
+ 22	HP	0x4	0x7	0	0x16
+ 20		0x3	0x5	0	0x16
+ 17		0x2	0x3	0	0x16
+ 14		0x2	0x2	0	0x16

Set_TxRxFallbackMode() command

`Set_TxRxFallbackMode(FallbackMode)` defines the operating mode to enter after a successful packet transmission or packet reception.

0	1
Opcode	FallbackMode[7:0]
w	w

byte 0 bits 7:0 **Opcode**: 0x93

byte 1 bits 7:0 **FallbackMode[7:0]**: Fall-back mode after successful packet transmission or packet reception
0x20: Standby mode entry (default)
0x30: Standby with HSE32 enabled mode entry
0x40: FS mode entry
Others: reserved

Set_CadParams() command

`Set_CadParams(NbCadSymbol,CadDetPeak,CadDetMin,CadExtiMode,Timeout)` allows the CAD configuration for LoRa packet types.

0	1	2	3	4	5	6	7
Opcode	NbCadSymbol[2:0]	CadDetPeak[7:0]	CadDetMin[7:0]	CadExitMode	Timeout[23:0]		
w	w	w	w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x88

byte 1 bits 7:3 Reserved, must be kept at reset value.

bits 2:0 **NbCadSymbol[2:0]**: Number of symbols used for CAD scan
0x0: 1 symbol
0x1: 2 symbols
0x2: 4 symbols
0x3: 8 symbols
0x4: 16 symbols
Others: reserved

byte 2 bits 7:0 **CadDetPeak[7:0]**: Used with CadDetMin[7:0] to correlate the LoRa symbol
See *Table 28*.

byte 3 bits 7:0 **CadDetMin[7:0]**: Used with CadDetPeak[7:0] to correlate the LoRa symbol
see *Table 28*.

byte 4 bits 7:1 Reserved, must be kept at reset value.

Bit 0 **CadExitMode**: defines the sub-GHz radio operating mode to enter after CAD scan is finished

0: Standby with RC 13 MHz mode entry after CAD, whatever is detected during the CAD scan

1: Standby with RC 13 MHz mode after CAD if no LoRa symbol is detected during the CAD scan

if a LoRa symbol is detected, the sub-GHz radio stays in Receive mode until a packet is received or until the CAD timeout is reached.

bytes 7:5 bits 23:0 **Timeout[23:0]**: CAD timeout = Timeout[23:0] x 15.625 µs

The CAD timeout is only used when a symbol is detected and CadExitMode = 1 (stay in Receive mode after a LoRa symbol detection).

0x000000 - 0xFFFFFF: timeout, resolution 15.625 µs

The correct values selected in the table below must be carefully tested to ensure a good detection at sensitivity level and to limit the number of false detections.

Table 28. Recommended CAD configuration settings

Spreading factor	cadDetMin	cadDetPeak	cadSymbolNum
SF7	10	22	2 symbols
SF8			
SF9		23	4 symbols
SF10		24	
SF11		25	
SF12		28	

Set_BufferBaseAddress() command

`Set_BufferBaseAddress(TxBaseAddr, RxBaseAddr)` sets the data buffer base address for the packet handling in TX and RX.

0	1	2
Opcode	TxBaseAddr[7:0]	RxBaseAddr[7:0]
w	w	w

byte 0 bits 7:0 **Opcode**: 0x8F

byte 1 bits 7:0 **TxBaseAddr[7:0]**: Tx base address offset relative to the sub-GHz RAM base address

byte 2 bits 7:0 **RxBaseAddr[7:0]**: Rx base address offset relative to the sub-GHz RAM base address

(G)FSK Set_ModulationParams() command

`Set_ModulationParams(Br, PulseShape, Bw, Fdev)` is used to configure the (G)FSK modulation parameters for the sub-GHz radio. Depending on the selected packet

type in `Set_PacketType()` sent prior to this function, the parameters for generic packets are interpreted as follows:

- Br and Fdev are used for the transmission and reception.
- Bw is used only for reception.
- PulseShape represents the Gaussian filter that can be used to filter the modulation stream at the transmitter.

0	1	2	3	4	5	6	7	8
Opcode	Br[23:0]			PulseShape[7:0]	Bw[7:0]	Fdev[23:0]		
w	w	w	w	w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x8B

bytes 3:1 bits 23:0 **Br[23:0]**: Bit rate
0x000000: reserved
0x000001 - 0xFFFFFF: Br = 32 x 32 MHz / bit rate

byte 4 bit 7:0 **PulseShape[7:0]**: Pulse shape
0x00: no filter applied
0x08:Gaussian BT 0.3
0x09: Gaussian BT 0.5
0x0A: Gaussian BT 0.7
0x0B: Gaussian BT 1.0
Others: no filter applied

bytes5 bit 7:0 **Bw[7:0]**: Bandwidth
0x1F: BW4 4.8 kHz DSB
0x17: BW5 5.8 kHz DSB
0x0F: BW7 7.3 kHz DSB
0x1E: BW9 9.7 kHz DSB
0x16: BW11 11.7 kHz DSB
0x0E: BW14 14.6 kHz DSB
0x1D: BW19 19.5 kHz DSB
0x15: BW23 23.4 kHz DSB
0x0D: BW29 29.3 kHz DSB
0x1C: BW39 39.0 kHz DSB
0x14: BW46 46.9 kHz DSB
0x0C: BW58 58.6 kHz DSB
0x1B: BW78 78.2 kHz DSB
0x13: BW93 93.8 kHz DSB
0x0B: BW117 117.3 kHz DSB
0x1A: BW156 156.2 kHz DSB
0x12: BW187 187.2 kHz DSB
0x0A: BW234 234.3 kHz DSB
0x19: BW312 312.0 kHz DSB
0x11: BW373 373.6 kHz DSB
0x09: BW467 467.0 kHz DSB
Others: reserved

bytes 8:6 bit 23:0 **Fdev[23:0]**: Frequency deviation
0x000000 - 0xFFFFFF: Fdev = Frequency deviation x 2^{25} / 32 MHz

LoRa Set_ModulationParams() command

`Set_ModulationParams(Sf, Bw, Cr, Ldro)` is used to configure the LoRa modulation parameters for the sub-GHz radio. Depending on the selected packet type in `Set_PacketType()` sent prior to this function, the parameters for LoRa are interpreted as below.

0	1	2	3	4
Opcode	Sf[3:0]	Bw[7:0]	Cr[2:0]	Ldro
w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x8B

byte 1 bits 7:4 Reserved, must be kept at reset value.

bits 3:0 **Sf[3:0]**: Spreading factor
0x5: Spreading factor 5
0x6: Spreading factor 6
0x7: Spreading factor 7
0x8: Spreading factor 8
0x9: Spreading factor 9
0xA: Spreading factor 10
0xB: Spreading factor 11
0xC: Spreading factor 12
Others: reserved

byte 2 bits 7:0 **Bw[7:0]**: Bandwidth
0x00: bandwidth 7 (7.81 kHz)
0x08: bandwidth 10 (10.42 kHz)
0x01: bandwidth 15 (15.63 kHz)
0x09: bandwidth 20 (20.83 kHz)
0x02: bandwidth 31 (31.25 kHz)
0x0A: bandwidth 41 (41.67 kHz)
0x03: bandwidth 62 (62.50 kHz)
0x04: bandwidth 125 (125 kHz)
0x05: bandwidth 250 (250 kHz)
0x06: bandwidth 500 (500 kHz)
Others: reserved

byte 3 bits 7:3 Reserved, must be kept at reset value.

bits 2:0 **Cr[2:0]**: Forward error correction coding rate
0x0: no forward error correction coding rate 4/4
0x1: forward error correction coding rate 4/5
0x2: forward error correction coding rate 4/6
0x3:forward error correction coding rate 4/7
0x4: forward error correction coding rate 4/8
Others: reserved

byte 4 bits 7:1 Reserved, must be kept at reset value.

bit 0 **Ldro**: low data rate optimization enable
0: low data rate optimization disabled
1: low data rate optimization enabled

BPSK Set_ModulationParams() command

`Set_ModulationParams(Br, PulseShape)` is used to configure the BPSK modulation parameters for the sub-GHz radio. Depending on the selected packet type in `Set_PacketType()` sent prior to this function, the parameters for BPSK packets are interpreted as follows:

- For BPSK packet type, the BitRate is used for the transmission.
- PulseShape represents the Gaussian filter that can be used to filter the modulation stream at the transmitter.

0	1	2	3	4
Opcode	Br[23:0]			PulseShape[7:0]
w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x8B

bytes 3:1 bits 23:0 **Br[23:0]**: bit rate
0x000000: reserved
0x000001 - 0xFFFFFF: Br = 32 x 32 MHz / bit rate
0x1A0AAA: 600 bit/s
0x9C4000: 100 bit/s

byte 4 bits 7:0 **PulseShape[7:0]**: Pulse shape
0x16: Gaussian BT 0.5
Others: reserved

Generic Set_PacketParams() command

`Set_PacketParams(PbLength,PbDetLength,SynchWordLength,AddrComp, PktType,PayloadLength,CrcType,Whitening)` is used to configure the packet handling for the sub-GHz radio. When the generic packet is selected with packet type in `Set_PacketType()` sent prior to this function, the parameters are interpreted as below.

0	1	2	3	4	5	6	7	8	9
Opcode	PbLength[15:0]		PbDetLength[2:0]	SyncWordLength[6:0]	AddrComp[1:0]	PktType	PayloadLength[7:0]	CrcType[2:0]	Whitening
w	w	w	w	w	w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x8C.

bytes 2:1 bits 15:0 **PbLength[15:0]**: Preamble length in number of symbols
0x0000: reserved
0x0001 - 0xFFFF: 1 to 65535 symbols

byte 3 bits 7:3 Reserved, must be kept at reset value.

bits 2:0 **PbDetLength[2:0]**: Preamble detection length in number of bit symbols
0x0: preamble detection disabled
0x4: 8-bit preamble detection
0x5: 16-bit preamble detection
0x6: 24-bit preamble detection
0x7: 32-bit preamble detection
Others: reserved

byte 4 bit 7 Reserved, must be kept at reset value.

bits: 6:0 **SyncWordLength[6:0]**: Synchronization word length in number of bit symbols
0x00 - 0x40: 0 to 64-bit synchronization word (synchronization word data defined in SUBGHZ_GSYNCR[0:7])
Others: reserved

byte 5 bits 7:2 Reserved, must be kept at reset value.

bits: 1:0 **AddrComp[1:0]**: Address comparison/filtering
0x0: address comparison/filtering disabled
0x1: address comparison/filtering on node address
0x2: address comparison/filtering on node and broadcast addresses
Others: reserved

byte 6 bits 7:1 Reserved, must be kept at reset value.

bit 0 **PktType**: Packet type definition
0: Fixed payload length and header field not added to packet
1: Variable payload length and header field added to packet

byte 7 bits 7:0 **PayloadLength[7:0]**: Payload length in number of bytes
0x00- 0xFF: 0 to 255 bytes

byte 8 bits 7:3 Reserved, must be kept at reset value.

bits 2:0 **CrcType[2:0]**: CRC type definition
The CRC initialization value is provided in SUBGHZ_GCRCINIRL and SUBGHZ_GCRCINIRH. The polynomial is defined in SUBGHZ_GCRCPOLRL and SUBGHZ_GCRCPOLRH.
0x0: 1-byte CRC
0x1: no CRC
0x2: 2-byte CRC
0x4: 1-byte inverted CRC
0x6: 2-byte inverted CRC
Others: reserved

byte 9 bits 7:1 Reserved, must be kept at reset value.

bit 0 **Whitening**: Whitening enable
The whitening initial value is provided in WHITEINI[8:0].
0: Whitening disabled
1: Whitening enabled

LoRa Set_PacketParams() command

`Set_PacketParams(PbLength, HeaderType, PayloadLength, CrcType, InvertIQ)` is used to configure the packet handling for the sub-GHz radio. Depending on the selected packet type in `Set_PacketType()` sent prior to this function, the parameters are interpreted as below.

0	1	2	3	4	5	6
Opcode	PbLength[15:0]		HeaderType	PayloadLength[7:0]	CrcType	InvertIQ
w	w	w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x8C.

bytes 2:1 bits 15:0 **PbLength[15:0]**: Preamble length in number of symbols
0x0000: reserved
0x0001 - 0xFFFF: 1 to 65535 symbols

byte 3 bits 7:1 Reserved, must be kept at reset value.

bit 0 **HeaderType**: Header type definition
0: explicit header for variable length payload
1: implicit header for fixed length payload

byte 4 bits 7:0 **PayloadLength[7:0]**: Payload length in number of bytes
0x00- 0xFF: 0 to 255 bytes

byte 5 bits 7:1 Reserved, must be kept at reset value.

bit 0 **CrcType** CRC enable
0: CRC disabled
1: CRC enabled

byte 6 bits 7:1 Reserved, must be kept at reset value.

bit 0 **InvertIQ**: IQ setup
0: standard IQ setup
1: inverted IQ setup

BPSK Set_PacketParams() command

`Set_PacketParams(PayloadLength)` is used to configure the packet handling for the sub-GHz radio. When the BPSK packet is selected with packet type in `Set_PacketType()` sent prior to this function, the parameters are interpreted as below.

0	1
Opcode	PayloadLength[7:0]
w	w

byte 0 bits 7:0 **Opcode**: 0x8C.

byte 1 bits 7:0 **PayloadLength[7:0]**: BPSK packet (payload) length, including preamble, synch word, device id and CRC, in number of bytes
0x00- 0xFF: 0 to 255 bytes

LoRa Set_LoRaSymbTimeout() command

`Set_LoRaSymbTimeout(SymbNum)` is used to configure the number of LoRa symbols to be received before starting the reception of a LoRa packet.

0	1
Opcode	SymbNum[7:0]
w	w

byte 0 bits 7:0 **Opcode**: 0xA0.

byte 1 bits 7:0 **SymbNum[7:0]**: Number of LoRa symbols before timeout IRQ is generated if end of preamble has not been detected
0x0: No detection of end of preamble
Others: When setting SymbNum to a value ≠ 0, the modem counts the number of received chirp after the first LoRa chirp is detected. The modem checks that the end of preamble is detected before the SymbNum number of LoRa symbol is obtained after the first chirp is detected. If this is not the case, a timeout is generated.

Note: In LoRa packet type, when entering the Receive mode and SymbNum[7:0] = 0x0, the modem locks as soon as a single LoRa symbol is detected. This may lead to a false detection. To avoid false detections, the number of LoRa symbols to be detected can be increased.

4.8.5 Communication status information commands

Get_Status() command

`Get_Status(Status)` can be issued at any time.

0	1
Opcode	Status[7:0]
w	r

byte 0 bits 7:0 **Opcode**: 0xC0

byte 1 bit 7 Reserved, must be kept at reset value.

bits 6:4 **Status_Mode[2:0]** sub-GHz radio operating mode
0x2: Standby mode with RC 13 MHz
0x3: Standby mode with HSE32
0x4: FS mode
0x5: RX mode
0x6: TX mode
Others: reserved

bits 3:1 **Status_CmdStatus[2:0]**: Command status
0x2: data available to host (packet received successfully and data can be retrieved)
0x3:command time out (command took too long to complete triggering a sub-GHz radio watchdog timeout)
0x4:command processing error (invalid opcode or incorrect number of parameters)
0x5:command execution failure (command successfully received but cannot be executed at this time, requested operating mode cannot be entered or requested data cannot be sent)
0x6: transmit command completed (current packet transmission completed)
Others: reserved

bit 0 Reserved, must be kept at reset value.

Get_RxBufferStatus() command

`Get_RxBufferStatus(Status, RxPayloadLength, RxBufferPointer)` returns the sub-GHz radio status, the length of the last received packet (RxPayloadLength) and the buffer address of the first received payload byte (RxBufferPointer).

0	1	2	3
Opcode	Status[7:0]	RxPayloadLength[7:0]	RxBufferPointer[7:0]
w	r	r	r

byte 0 bits 7:0 **Opcode**: 0x13.

byte 1 bits 7:0 **Status[7:0]**: see *Get_Status() command*.

byte 2 bits 7:0 **RxPayloadLenght[7:0]**: indicate the number of bytes received in the last received packet

byte 3 bits 7:0 **RxBufferPointer[7:0]**: indicates the offset in the RAM data buffer where the first byte of the last received packet is stored

(G)FSK Get_PacketStatus() command

`Get_PacketStatus(Status, RxStatus, RssiSync, RssiAvg)` returns information on the last received packet. Depending on the selected packet type in `Set_PacketType()` sent prior to this function, the parameters for generic packets are interpreted as below.

0	1	2	3	4
Opcode	Status[7:0]	RxStatus[7:0]	RssiSync[7:0]	RssiAvg[7:0]
w	r	r	r	r

byte 0 bits 7:0 **Opcode**: 0x14

byte 1 bits 7:0 **Status[7:0]**: see *Get_Status() command*

byte 2 bit 7 **RxStatus_PreambleErr**: Set when a preamble error occurred

bit 6 **RxStatus_SyncErr**: Set when a synchronization error occurred

bit 5 **RxStatus_AdrsErr**: Set when an address error occurred

bit 4 **RxStatus_CrcErr**: Set when a crc error occurred

bit 3 **RxStatus_LengthErr**: Set when a length error occurred

bit 2 **RxStatus_AbortErr**: Set when an abort error occurred

bit 1 **RxStatus_PktReceived**: Set when a packet is received

bit 0 **RxStatus_PktSent**: Set when a packet has been sent

byte 3 bits 7:0 **RssiSync[7:0]**: RSSI level when the synchronization address is detected
Signal power = -RssiSync / 2 (in dBm)

byte 4 bits 7:0 **RssiAvg[7:0]**: Average RSSI level over the received packet
Signal power = -RssiAvg / 2 (in dBm)

LoRa Get_PacketStatus() command

`Get_PacketStatus(Status, RssiPkt, SnrPkt, SignalRssiPkt)` returns information on the last received packet. Depending on the selected packet type in `Set_PacketType()` sent prior to this function, the parameters for LoRa packets are interpreted as below.

0	1	2	3	4
Opcode	Status[7;0]	RssiPkt[7:0]	SnrPkt[7:0]	SignalRssiPkt
w	r	r	r	r

byte 0 bits 7:0 **Opcode**: 0x14

byte 1 bits 7:0 **Status[7:0]**: see *Get_Status() command*

byte 2 bits 7:0 **RssiPkt[7:0]**: Average RSSI level over the received packet
Signal power = - RssiPkt / 2 (in dBm)

byte 3 bits 7:0 **SnrPkt[7:0]**: Estimation of SNR over the received packet
SNR = SnrPkt / 4 (in dB)

byte 4 bits 7:0 **SignalRssiPkt[7:0]**: Estimation of RSSI level of the LoRa signal after despreading
Signal power = - SignalRssiPkt / 2 (in dBm)

Get_RssiInst() command

`Get_RssiInst(Status, RssiInst)` returns the instantaneous signal strength during packet reception.

0	1	2
Opcode	Status[7:0]	RssiInst[7:0]
w	r	r

byte 0 bits 7:0 **Opcode**: 0x15

byte 1 bits 7:0 **Status[7:0]**: see *Get_Status() command*

byte 2 bits 7:0 **RssiInst[7:0]**: instantaneous RSSI level at the reception time
Signal power = - RssiInst / 2 (in dBm)

(G)FSK Get_Stats() command

`Get_Stats(Status, NbPktReceived, NbPktCrcError, NpPktLengthError)` returns statistics information on the last received packet. Depending on the selected packet type in `Set_PacketType()` sent prior to this function, the parameters for generic packets are interpreted as below.

0	1	2	3	4	5	6	7
Opcode	Status[7:0]	NbPktReceived[15:0]		NbPktCrcError[15:0]		NbPktLengthError[15:0]	
w	r	r	r	r	r	r	r

byte 0 bits 7:0 **Opcode**: 0x10

byte 1 bits 7:0 **Status[7:0]**: see *Get_Status() command*

bytes 3:2 bits 15:0 **NbPktReceived[15:0]**: Number of packets received

bytes 5:4 bits 15:0 **NbPktCrcError[15:0]**: Number of packets received with a payload CRC error

bytes 7:6 bits 15:0 **NbPktLengthError[15:0]**: Number of packets received with a payload length error

LoRa Get_Stats() command

`Get_Stats(Status, NbPktReceived, NbPktCrcError, NpPktHeaderError)` returns statistics information on the last received packet. Depending on the selected packet type in `Set_PacketType()` sent prior to this function, the parameters for LoRa packets are interpreted as below.

0	1	2	3	4	5	6	7
Opcode	Status[7:0]	NbPktReceived[15:0]		NbPktCrcError[15:0]		NbPktHeaderError[15:0]	
w	r	r	r	r	r	r	r

byte 0 bits 7:0 **Opcode**: 0x10

byte 1 bits 7:0 **Status[7:0]**: see *Get_Status() command*

bytes 3:2 bits 15:0 **NbPktReceived[15:0]**: Number of packets received

bytes 5:4 bits 15:0 **NbPktCrcError[15:0]**: Number of packets received with a payload CRC error.

bytes 7:6 bits 15:0 **NbPktHeaderError[15:0]**: Number of packets received with a header CRC error

Reset_Stats() command

`Reset_Stats(0x00,0x00,0x00,0x00,0x00,0x00)` resets the received packet statistics as reported in `Get_Stats()` (NbPktReceived, NbPktCrcError, NbPktlengthError and NbPktHeaderError).

0	1	2	3	4	5	6
Opcode	0x00	0x00	0x00	0x00	0x00	0x00
w	w	w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x0

byte 1 bits 7:0 **0x0**

byte 2 bits 7:0 **0x0**

byte 3 bits 7:0 **0x0**

byte 4 bits 7:0 **0x0**

byte 5 bits 7:0 **0x0**

byte 6 bits 7:0 **0x0**

4.8.6 IRQ interrupt commands

There are three IRQ interrupts that can be mapped to several sub-GHz radio interrupt sources. The source of an interrupt is determined by reading the device status. Interrupts are cleared using `Clr_IrqStatus()`.

There are 10 possible interrupt sources used depending on the packet type and operating mode. Each of these interrupt sources can be enabled or masked and mapped on any of the IRQ interrupts.

A set of commands is used to configure and control the IRQ sources and interrupt generation

Table 29. IRQ bit mapping and definition

Bit	Source	Description	Packet type	Operation
0	TxDone	Packet transmission finished	LoRa and GFSK	Tx
1	RxDone	Packet reception finished	LoRa and GFSK	Rx
2	PreambleDetected	Preamble detected	LoRa and GFSK	Rx
3	SyncDetected	Synchronization word valid	GFSK	Rx
4	HeaderValid	Header valid	LoRa	Rx
5	HeaderErr	Header CRC error	LoRa	Rx

Table 29. IRQ bit mapping and definition (continued)

Bit	Source	Description	Packet type	Operation
6	Err	preamble, syncword, address, CRC or length error	GFSK	Rx
	CrcErr	CRC error	LoRa	Rx
7	CadDone	Channel activity detection finished	LoRa	Cad
8	CadDetected	Channel activity detected	LoRa	Cad
9	Timeout	RX or TX timeout	LoRa and GFSK	Rx and Tx
15:10	Not applicable	Reserved	Not applicable	

Cfg_DioIrq() command

`Cfg_DioIrq(IrqMask, Irq1Mask, Irq2Mask, Irq3Mask)` allows interrupts to be masked and mapped on the IRQ lines.

0	1	2	3	4	5	6	7	8
Opcode	IrqMask[15:0]		Irq1Mask[15:0]		Irq2Mask[15:0]		Irq3Mask[15:0]	
w	w	w	w	w	w	w	w	w

byte 0 bits 7:0 **Opcode**: 0x08

bytes 2:1 bits 15:0 **IrqMask[15:0]**: Global interrupt enable
See *Table 29* for interrupt bit map definition. For each bit:
0: IRQ disabled
1: IRQ enabled

bytes 4:3 bits 15:0 **Irq1Mask[15:0]**: IRQ1 line Interrupt enable
0: interrupt on IRQ1 line disable
1: interrupt on IRQ1 line enabled

bytes 6:5 bits 15:0 **Irq2Mask[15:0]**: IRQ2 line Interrupt enable
0: interrupt on IRQ2 line disable
1: interrupt on IRQ2 line enabled

bytes 8:7 bits 15:0 **Irq3Mask[15:0]**: IRQ3 line Interrupt enable
0: interrupt on IRQ3 line disable
1: interrupt on IRQ3 line enabled

Get_IrqStatus() command

`Get_IrqStatus(Status, IrqStatus)` returns the IRQ status.

0	1	2	3
Opcode	Status[7:0]	IrqStatus[15:0]	
w	r	r	r

byte 0 bits 7:0 **Opcode**: 0x12

byte 1 bits 7:0 **Status[7:0]**: see *Get_Status() command*

bytes 3:2 bits 15:0 **IrqStatus[15:0]**: interrupt pending status information
See *Table 29* for interrupt bit map definition. For each bit:
0: IRQ not pending
1: IRQ pending

Clr_IrqStatus() command

`Clr_IrqStatus(ClrIrq)` clears the IRQ status flags (IrqStatus[15:0]).

0	1	2
Opcode	ClrIrq[15:0]	
w	w	w

byte 0 bits 7:0 **Opcode**: 0x02

bytes 2:1 bits 15:0 **ClrIrq[15:0]**: Clear interrupt status
See *Table 29* for interrupt bit map definition. For each bit:
0: no action
1: IRQ pending status flag cleared

4.8.7 Miscellaneous commands

Calibrate() command

`Calibrate(CalibCfg)` allows one or several blocks to be calibrated at any time when in Standby mode. The blocks to calibrate are defined by CalibCfg parameter. When the calibration is ongoing, BUSY is set. A falling edge on BUSY indicates the end of all enabled calibrations.

0	1
Opcode	CalibCfg[7:0]
w	w

byte 0 bits 7:0 **Opcode**: 0x89

byte 1

bit 7 Reserved, must be kept at reset value.

bit 6 **CalibCfg_Image**: Image calibration
- 0: Image calibration disabled
- 1: Image calibration enabled

bit 5 **CalibCfg_AdcBulkP**: RF-ADC bulk P calibration
- 0: RF-ADC bulk P calibration disabled
- 1: RF-ADC bulk P calibration enabled

bit 4 **CalibCfg_AdcBulkN**: RF-ADC bulk N calibration
- 0: RF-ADC bulk N calibration disabled
- 1: RF-ADC bulk N calibration enabled

bit 3 **CalibCfg_AdcPulse**: RF-ADC pulse calibration
- 0: RF-ADC pulse calibration disabled
- 1: RF-ADC pulse calibration enabled

bit 2 **CalibCfg_Pll**: RF-PLL calibration
- 0: RF-PLL calibration disabled
- 1: RF-PLL calibration enabled

bit 1 **CalibCfg_RC13M**: Sub-GHz radio RC 13 MHz calibration
- 0: Sub-GHz radio RC 13 MHz calibration disabled
- 1: Sub-GHz radio RC 13 MHz calibration enabled

bit 0 **CalibCfg_RC64K**: Sub-GHz radio RC 64 kHz calibration
- 0: Sub-GHz radio RC 64 kHz calibration disabled
- 1: Sub-GHz radio RC 64 kHz calibration enabled

CalibrateImage() command

`CalibrateImage(CalFreq1, CalFreq2)` allows the image to be calibrated at given frequencies, at any time when in Standby mode. The calibration frequencies are defined by CalFreq1 and CalFreq2. When the calibration is ongoing, BUSY is set. A falling edge on BUSY indicates the end of calibration. Once performed, the calibration is valid for all frequencies between the two selected calibration frequencies. Any ISM band can be covered by selecting the applicable frequency for CalFreq1 and CalFreq2

Table 30. Image calibration for ISM bands

ISM band [MHz]	CalIFreq1	CalIFreq2
430 - 440	0x6B	0x6F
470 - 510	0x75	0x81
779 - 787	0xC1	0xC5
863 - 870	0xD7	0xDB
902 - 928	0xE1 (default)	0xE9 (default)

By default, the image calibration is made in the band 902 - 928 MHz. Nevertheless it is possible to request a new calibration at other frequencies.

0	1	2
Opcode	CalFreq1[7:0]	CalFreq2[7:0]
w	w	w

byte 0 bits 7:0 **Opcode**: 0x98

byte 1 bits 7:0 **CalFreq1[7:0]**: Lower frequency of the band to calibrate (see *Table 30*)

byte 2 bits 7:0 **CalFreq2[7:0]**: Higher frequency of the band to calibrate (see *Table 30*)

The calibration frequencies are computed as follows:

$Calibration_{freq}$ = CalFreq * 4 MHz where CalFreq1 ≤ CalFreq2

Example: 0x6B = 428 MHz.

When CalFreq1 = CalFreq2, the image calibration is done at a single frequency.

For frequencies between CalFreq1 and CalFreq2, the calibration coefficient is linearly interpolated from the values obtained during the image calibration at CalFreq1 and CalFreq2.

For frequencies ≤ CalFreq1, the coefficient obtained during the image calibration at CalFreq1 is used.

For frequencies ≥ CalFreq2, the coefficient obtained during the image calibration at CalFreq2 is used.

Set_RegulatorMode() command

`Set_RegulatorMode(RegMode)` allows the sub-GHz radio supply operation mode selection, between LDO mode (default) and SMPS mode, when in Standby with HSE32 and Active modes.

0	1
Opcode	RegMode
w	w

byte 0 bits 7:0 **Opcode**: 0x96

byte 1 bits 7:1 Reserved, must be kept at reset value.

bit 0 **RegMode**: Regulator mode selection when sub-GHz radio is active
0: LDO mode
1: SMPS mode used in Standby with HSE32, FS, RX and TX modes

Note: For the different CPU operating modes, the LDO or SMPS mode is controlled by the SMPSEN bit in PWR control register 5 (PWR_CR5). When the sub-GHz radio or CPU selects the SMPS mode, the LDO mode selection is discarded.

Get_Error() command

`Get_Error(Status, OpError)` returns the sub-GHz radio operational errors.

0	1	2	3
Opcode	Status[7:0]	OpError[15:0]	
w	r	r	r

byte 0 bits 7:0 **Opcode**: 0x17

byte 1 bits 7:0 **Status[7:0]**: see *Get_Status() command*

bytes 3:2 bits 15:9 Reserved, must be kept at reset value.

bit 8 **OpError_PaRampError**: PA ramping failed

bit 7 Reserved, must be kept at reset value.

bit 6 **OpError_PllLockError**: RF-PLL locking failed

bit 5 **OpError_XoscStartError**: HSE32 clock startup failed

bit 4 **OpError_ImageCalibrationError**: Image calibration failed

bit 3 **OpError_AdcCalibrationError**: RF-ADC calibration failed

bit 2 **OpError_PllCalibrationError**: RF-PLL calibration failed

bit 1 **OpError_RC13MCalibrationError**: Sub-GHz radio RC 13 MHz oscillator calibration failed

bit 0 **OpError_RC64KCalibrationError**: Sub-GHz radio RC 64 kHz oscillator calibration failed

Clr_Error() command

`Clr_Error(0x00)` clears all recorded sub-GHz radio errors, as reported in `Get_Error()` (RC64KCalibrationError, RC13MCalibrationError, PllCalibrationError, AdcCalibrationError, ImageCalibrationError, XscoStartError, PllLockError and PaRampError).

0	1
Opcode	0x00
w	w

byte 0 bits 7:0 **Opcode**: 0x07

byte 1 bits 7:0 0x00

4.8.8 Set_TcxoMode command

`Set_TcxoMode(RegTcxoTrim,Timeout)` is used to configure the TCXO and HSE32 ready timeout.

Table 31. Command format Set_TcxoMode()

Byte 0	Byte 1	Byte 2-4
Opcode = 0x97	RegTcxoTrim[2:0]	Timeout[23:0]

The definition of RegTcxoTrim and Timeout bytes is given in the table below.

Table 32. RegTcxoTrim and Timeout bytes definition

Byte 1 [7:3]	Byte 1 RegTcxoTrim[2:0[(1)]] (V)	Byte 2-4 [23:0] timeout
Reserved	0x0 = 1.6	0x000000 = timeout disabled
	0x1 = 1.7	Other = timeout enabled[(2)]
	0x2 = 1.8	-
	0x3 = 2.2	-
	0x4 = 2.4	-
	0x5 = 2.7	-
	0x6 = 3.0	-
	0x7 = 3.3	-

1. To use V_{DDTCXO}, the V_{DDRF} supply must be at least + 200 mV higher than the selected RegTcxoTrim voltage level.
2. Maximum time the system waits for the HSE32 clock to be ready, before HSEStartErr is set.

The time-out duration is computed by the following formula:

Time-out duration = Timeout x 15.625 µs (maximum time-out duration = 262.14 s)

4.8.9 Sub-GHz radio commands overview

The sub-GHz radio commands are mapped as 8-bit addressable SPI commands, as described in the table below:

Table 33. Sub-GHz radio SPI commands overview

Command		Opcode	Parameters
`CalibratImage()`		0x98	CalFreq1, CalFreq2
`Calibrate()`		0x89	CalibCfg
`Cfg_DioIrq()`		0x08	IrqMask, Irq1Mask, Irq2Mask, Irq3Mask
`Clr_Error()`		0x07	0x00
`Clr_IrqStatus()`		0x02	ClrIrq
`Get_Error()`		0x17	Status, OpError
`Get_IrqStatus()`		0x12	Status, IrqStatus
`Get_PacketStatus()`	FSK	0x14	Status, RxStatus, RssiSync, RssiAvg
	LoRa		Status. RssiPkt, SnrPkt, SignalRssiPkt
`Get_PacketType()`		0x11	Status, PktType
`Get_RssiInst()`		0x15	Status, RssiInst
`Get_RxBufferStatus()`		0x13	Status, RxPayloadLength, RxBufferPointer
`Get_Stats()`	FSK	0x10	Status, NbPktReceived, NbPktCrcError, NbPktLengthError
	LoRa		Status, NbPktReceived, NbPktCrcError, NbPktHeaderError

Table 33. Sub-GHz radio SPI commands overview (continued)

Command		Opcode	Parameters
Get_Status()		0xC0	Status
Read_Buffer()		0x1E	Offset, Status, Data0, Data1, ..., Datan
Read_Register()		0x1D	Addr, Status, Data0, Data1, ..., Datan
Reset_Stats()		0x00	0x00, 0x00, 0x00, 0x00, 0x00, 0x00
Set_BufferBaseAddress()		0x8F	TxBaseAddr, RxBaseAddr
Set_Cad()		0xC5	-
Set_CadParams()		0x88	NbCadSymbol, CadDetPeak, CadDetMin, CadExitMode, Timeout
Set_Fs()		0xC1	-
Set_LoRaSymbTimeout()		0xA0	SymbNum
Set_ModulationParams()	FSK	0x8B	Br, PulseShape, Bw, Fdev
	LoRa		Sf, Bw, Cr, Ldro
	BPSK		Br, PulseShape
Set_PaConfig()		0x95	PaDutyCycle, HpMax, HpSel, 0x01
Set_PacketParams()	Generic	0x8C	PbLength, PdDetLength, SyncWordLength, AddrComp, PktType, PayloadLength, CrcType, Whitening
	LoRa		PbLength, HeaderType, PayloadLength, CrcType, InvertIQ
	BPSK		PayloadLength
Set_PacketType()		0x8A	PktType
Set_RegulatorMode()		0x96	RegMode
Set_RfFrequency()		0x86	RfFreq
Set_Rx()		0x82	Timeout
Set_RxDutyCycle()		0x94	RxPeriod, SleepPeriod
Set_TxRxFallbackMode()		0x93	FallbackMode
Set_Sleep()		0x84	SleepCfg
Set_Standby()		0x80	StandbyCfg
Set_StopRxTimerOnPreamble()		0x9F	RxTimeoutStop
Set_TcxoMode()		0x97	RegTcxoTrim, Timeout
Set_Tx()		0x83	Timeout
Set_TxContinuousWave()		0xD1	-
Set_TxContinuousPreamble()		0xD2	-
Set_TxParams()		0x8E	Power, RampTime
Write_Buffer()		0x0E	Offset, Data0, Data1, ..., Datan
Write_Register()		0x0D	Addr, Data0, Data1, ..., Datan

4.9 Sub-GHz radio application configuration

The sub-GHz radio is controlled via the SPI command interface. The following sections describe the basic sequence for some sub-GHz radio operations.

After releasing the sub-GHz radio reset and waking it up with sub-GHz radio SPI NSS, the sub-GHz radio automatically performs a calibration and enters Standby mode. When the sub-GHz radio is already active, the Standby mode can be requested by `Set_Standby()`. Once BUSY goes low, the sub-GHz radio can be set in Active mode.

4.9.1 Basic sequence for LoRa, (G)MSK and (G)FSK transmit operation

The sub-GHz radio can be set in LoRa, (G)MSK or (G)FSK transmit operation mode with the following steps:

1. Define the location of the transmit payload data in the data buffer, with `Set_BufferBaseAddress()`.
2. Write the payload data to the transmit data buffer with `Write_Buffer()`.
3. Select the packet type (generic or LoRa) with `Set_PacketType()`.
4. Define the frame format with `Set_PacketParams()`.
5. Define synchronization word in the associated packet type SUBGHZ_xSYNCR(n) with `Write_Register()`.
6. Define the RF frequency with `Set_RfFrequency()`.
7. Define the PA configuration with `Set_PaConfig()`.
8. Define the PA output power and ramping with `Set_TxParams()`.
9. Define the modulation parameters with `Set_ModulationParams()`.
10. Enable TxDone and timeout interrupts by configuring IRQ with `Cfg_DioIrq()`.
11. Start the transmission by setting the sub-GHz radio in TX mode with `Set_Tx()`. After the transmission is finished, the sub-GHz radio enters automatically the Standby mode.
12. Wait for sub-GHz radio IRQ interrupt and read interrupt status with `Get_IrqStatus()`:
    a) On a TxDone interrupt, the packet is successfully sent
    b) On a timeout interrupt, the transmission is timeout.
13. Clear interrupt with `Clr_IrqStatus()`.
14. Optionally, send a `Set_Sleep()` command to force the sub-GHz radio in Sleep mode.

4.9.2 Basic sequence for LoRa and (G)FSK receive operation

The sub-GHz radio can be set in LoRa or (G)FSK receive operation mode with the following steps:

1. Define the location where the received payload data must be stored in the data buffer, with `Set_BufferBaseAddress()`.
2. Select the packet type (generic or LoRa) with `Set_PacketType()`.
3. Define the frame format with `Set_PacketParams()`.
4. Define synchronization word in the associated packet type SUBGHZ_xSYNCR(n) with `Write_Register()`.
5. Define the RF frequency with `Set_RfFrequency()`.
6. Define the modulation parameters with `Set_ModulationParams()`.
7. Enable RxDone and timeout interrupts by configuring IRQ with `Cfg_DioIrq()`.
8. Start the receiver by setting the sub-GHz radio in RX mode with `Set_Rx()`:
   - When in continuous receiver mode, the sub-GHz radio remains in RX mode to look for packets until stopped with `Set_Standby()`.
   - In single mode (with or without timeout), when the reception is finished, the sub-GHz radio enters automatically the Standby mode.
   - In listening mode, the sub-GHz radio repeatedly switches between RX single with timeout mode and Sleep mode.
9. Wait for sub-GHz radio IRQ interrupt and read the interrupt status with `Get_IrqStatus()`:
   a) On a RxDone interrupt, a packet is received:
   - Check received packet error status (header error, crc error) with `Get_IrqStatus()`.
   - When a valid packet is received, read the receive start buffer pointer and received payload length with `Get_RxBufferStatus()`.
   - Read the received payload data from the receive data buffer with `Read_Buffer()`.

   b) On a timeout interrupt, the reception is timed out.
10. Clear interrupts with `Clr_IrqStatus()`.
11. Optionally, send a `Set_Sleep()` command to force the sub-GHz radio in Sleep mode.

4.9.3 Basic sequence for BPSK transmit operation

The sub-GHz radio can be set in BPSK transmit operation mode by the following steps:

1. Define the location of the transmit payload data in the data buffer, with `Set_BufferBaseAddress()`
2. Write the packet data (synchronization word, payload data) to the transmit data buffer with `Write_Buffer()`.
3. Select the packet type (BPSK) with `Set_PacketType()`.
4. Define the frame format with `Set_PacketParams()`.
5. Define the RF frequency with `Set_RfFrequency()`.
6. Define the PA configuration with `Set_PaConfig()`.
7. Define the PA output power and ramping with `Set_TxParams()`.
8. Define the modulation parameters with `Set_ModulationParams()`.
9. Enable TxDone and Timeout interrupts by configuring IRQ with `Cfg_DioIrq()`.
10. Start the transmission by setting the sub-GHz radio in TX mode with `Set_Tx()`. After the transmission is finished, the sub-GHz radio enters automatically the Standby mode.
11. Wait for sub-GHz radio IRQ interrupt and read interrupt status with `Get_IrqStatus()`:
    a) On a TxDone interrupt, the packet is successfully sent.
    b) On a timeout interrupt, the transmission is timed out.
12. Clear interrupt with `Clr_IrqStatus()`.
13. Optionally, send a `Set_Sleep()` command to force the sub-GHz radio in Sleep mode.

4.10 Sub-GHz radio registers

The sub-GHz radio peripheral registers can be accessed by sub-GHz radio command `Write_Register()` and `Read_Register()`.

4.10.1 Sub-GHz radio ramp-up MSB register (SUBGHZ_RAM_RAMPUPH)

Address offset: 0x0F0

Reset value: 0x00

7	6	5	4	3	2	1	0
RAMPUP[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RAMPUP[15:8]:** Ramp-up MSB bits

4.10.2 Sub-GHz radio ramp-up LSB register (SUBGHZ_RAM_RAMPUPL)

Address offset: 0x0F1

Reset value: 0x00

7	6	5	4	3	2	1	0
RAMPUP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RAMPUP[7:0]:** Ramp-up LSB bits

4.10.3 Sub-GHz radio ramp-down MSB register (SUBGHZ_RAM_RAMPDNH)

Address offset: 0x0F2

Reset value: 0x00

7	6	5	4	3	2	1	0
RAMPDN[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RAMPDN[15:8]:** Ramp-down MSB bits

4.10.4 Sub-GHz radio ramp-down LSB register (SUBGHZ_RAM_RAMPDNL)

Address offset: 0x0F3

Reset value: 0x00

7	6	5	4	3	2	1	0
RAMPDN[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RAMPDN[7:0]:** Ramp-down LSB bits

4.10.5 Sub-GHz radio frame limit MSB register (SUBGHZ_RAM_FRAMELIMH)

Address offset: 0x0F4

Reset value: 0x00

7	6	5	4	3	2	1	0
FRAMELIMH[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **FRAMELIMH[15:8]:** frame limit MSB bits

4.10.6 Sub-GHz radio frame limit LSB register (SUBGHZ_RAM_FRAMELIML)

Address offset: 0x0F5

Reset value: 0x00

7	6	5	4	3	2	1	0
FRAMELIML[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **FRAMELIML[7:0]:** frame limit LSB bits

4.10.7 Sub-GHz radio generic bit synchronization register (SUBGHZ_GBSYNCR)

Address offset: 0x6AC

Reset value: 0x00

This register must be cleared to 0x00 when using packet types other than LoRa.

7	6	5	4	3	2	1	0
Res.	SBITSYNCEN	RXDINV	BITSYNCDIS	Res.	Res.	Res.	Res.
	rw	rw	rw				

Bit 7 Reserved, must be kept at reset value.

Bit 6 **SBITSYNCEN:** LoRa simple bit synchronization enable
This bit must be cleared to 0 when using generic packet and BPSK type.
0: simple bit synchronization disabled
1: simple bit synchronization enabled

Bit 5 **RXDINV:** LoRa receive data inversion
This bit must be cleared to 0 when using generic packet and BPSK type.
0: receive data not inverted
1: receive data inverted

Bit 4 **BITSYNCDIS:** LoRa normal bit synchronization enable
This bit must be cleared to 0 when using generic packet and BPSK type.
0: normal bit synchronization enabled
1: normal bit synchronization disabled

Bits 3:0 Reserved, must be kept at reset value.

4.10.8 Sub-GHz radio generic CFO MSB register (SUBGHZ_GCFORH)

Address offset: 0x6B0

Reset value: 0x00

7	6	5	4	3	2	1	0
Res	Res	Res	Res	DEMOD_CFO[3:0]			
				r	r	r	r

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **DEMOD_CFO[3:0]:** actual frequency error from normalized value (MSB bits)

4.10.9 Sub-GHz radio generic CFO LSB register (SUBGHZ_GCFORL)

Address offset: 0x6B1

Reset value: 0x00

7	6	5	4	3	2	1	0
DEMOD_CFO[7:0]							
r	r	r	r	r	r	r	r

Bits 7:0 **DEMOD_CFO[7:0]:** actual frequency error from normalized value (LSB bits)

4.10.10 Sub-GHz radio generic packet control 1 register (SUBGHZ_GPKTCTL1R)

Address offset: 0x06B4

Reset value: 0x04

This register must be cleared to 0x00 when using packet types other than LoRa.

7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	PBDETON	PBDETLEN[1:0]	
					rw	rw	rw

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **PBDETON:** Preamble detection enable

Bits 1:0 **PBDETLEN:** Receiver preamble detection length
0b00: 8-bit preamble detection
0b01: 16-bit preamble detection
0b10: 24-bit preamble detection
0b11: 32-bit preamble detection

4.10.11 Sub-GHz radio generic packet control 1A register (SUBGHZ_GPKTCTL1AR)

Address offset: 0x6B8

Reset value: 0x21

7	6	5	4	3	2	1	0
Res.	Res.	SYNCDETEN	CONTTX	INFSEQSEL[1:0]		INFSQEQEN	WHITEINI[8]
		rw	rw	rw	rw	rw	rw

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **SYNCDETEN:** Generic packet synchronization word detection enable

Bit 4 **CONTTX:** Generic packet continuous transmit enable

Bits 3:2 **INFSEQSEL[1:0]:** Generic packet infinite sequence selection
00: preamble 0x5555
01: all zero 0x0000
10: all one 0xFFFF
11: PRBS9

Bit 1 **INFSQEQEN:** Generic packet infinite sequence enable

Bit 0 **WHITEINI[8]:** Generic packet whitening initial value MSB bit [8]

4.10.12 Sub-GHz radio generic whitening LSB register (SUBGHZ_GWHITEINIRL)

Address offset: 0x6B9

Reset value: 0x00

7	6	5	4	3	2	1	0
WHITEINI[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **WHITEINI[7:0]:** Generic packet whitening initial value LSB bits [7:0]

4.10.13 Sub-GHz radio generic payload length register (SUBGHZ_GRTXPLDLEN)

Address offset: 0x6BB

Reset value: 0x0F

7	6	5	4	3	2	1	0
RTXPLDLEN[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RTXPLDLEN[7:0]:** Payload length FIFO match value in Rx and Tx

4.10.14 Sub-GHz radio generic CRC initial MSB register (SUBGHZ_GCRCINIRH)

Address offset: 0x6BC

Reset value: 0x1D

7	6	5	4	3	2	1	0
CRCINI[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CRCINI[15:8]:** Generic packet CRC initial polynomial MSB bits [15:8]
These bits are used for CRC initialization.

4.10.15 Sub-GHz radio generic CRC initial LSB register (SUBGHZ_GCRCINIRL)

Address offset: 0x6BD

Reset value: 0x0F

7	6	5	4	3	2	1	0
CRCINI[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CRCINI[7:0]:** Generic packet CRC initial polynomial LSB bits [7:0]
These bits are used for CRC initialization.

4.10.16 Sub-GHz radio generic CRC polynomial MSB register (SUBGHZ_GCRCPOLRH)

Address offset: 0x6BE

Reset value: 0x10

7	6	5	4	3	2	1	0
CRCPOL[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CRCPOL[15:8]:** Generic packet CRC polynomial MSB bits [15:8]
These bits are used for CRC initialization.

4.10.17 Sub-GHz radio generic CRC polynomial LSB register (SUBGHZ_GCRCPOLRL)

Address offset: 0x6BF

Reset value: 0x21

7	6	5	4	3	2	1	0
CRCPOLI[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CRCPOLI[7:0]:** Generic packet CRC initial polynomial LSB bits [7:0]
These bits are used for CRC initialization.

4.10.18 Sub-GHz radio generic synchronization word control register 7 (SUBGHZ_GSYNCR7)

Address offset: 0x6C0

Reset value: 0x97

7	6	5	4	3	2	1	0
SYNCWORD[63:56]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[63:56]:** Eight byte of generic packet synchronization word

4.10.19 Sub-GHz radio generic synchronization word control register 6 (SUBGHZ_GSYNCR6)

Address offset: 0x6C1

Reset value: 0x23

7	6	5	4	3	2	1	0
SYNCWORD[55:48]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[55:48]:** Seventh byte of generic packet synchronization word

4.10.20 Sub-GHz radio generic synchronization word control register 5 (SUBGHZ_GSYNCR5)

Address offset: 0x6C2

Reset value: 0x52

7	6	5	4	3	2	1	0
SYNCWORD[47:40]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[47:40]:** Sixth byte of generic packet synchronization word

4.10.21 Sub-GHz radio generic synchronization word control register 4 (SUBGHZ_GSYNCR4)

Address offset: 0x6C3

Reset value: 0x25

7	6	5	4	3	2	1	0
SYNCWORD[39:32]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[39:32]:** Fifth byte of generic packet synchronization word

4.10.22 Sub-GHz radio generic synchronization word control register 3 (SUBGHZ_GSYNCR3)

Address offset: 0x6C4

Reset value: 0x56

7	6	5	4	3	2	1	0
SYNCWORD[31:24]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[31:24]** Fourth byte of generic packet synchronization word

4.10.23 Sub-GHz radio generic synchronization word control register 2 (SUBGHZ_GSYNCR2)

Address offset: 0x6C5

Reset value: 0x53

7	6	5	4	3	2	1	0
SYNCWORD[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[23:16]:** Third byte of generic packet synchronization word

4.10.24 Sub-GHz radio generic synchronization word control register 1 (SUBGHZ_GSYNCR1)

Address offset: 0x6C6

Reset value: 0x65

7	6	5	4	3	2	1	0
SYNCWORD[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[15:8]:** Second byte of generic packet synchronization word

4.10.25 Sub-GHz radio generic synchronization word control register 0 (SUBGHZ_GSYNCR0)

Address offset: 0x6C7

Reset value: 0x64

7	6	5	4	3	2	1	0
SYNCWORD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[7:0]:** First byte of generic packet synchronization word

4.10.26 Sub-GHz radio generic node address register (SUBGHZ_GNODEADR)

Address offset: 0x6CD

Reset value: 0x00

7	6	5	4	3	2	1	0
NODEADD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **NODEADD[7:0]:** Node address used in FSK mode register

4.10.27 Sub-GHz radio generic broadcast address register (SUBGHZ_GBCASTADDR)

Address offset: 0x6CE

Reset value: 0x00

7	6	5	4	3	2	1	0
BCASTADD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **BCASTADD[7:0]:** Broadcast address used in FSK mode register

4.10.28 Sub-GHz radio generic AFC register (SUBGHZ_GAFCR)

Address offset: 0x6D1

Reset value: 0x18

7	6	5	4	3	2	1	0
AFC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **AFC[7:0]:** Automatic frequency control register

4.10.29 Sub-GHz radio LoRa payload length register (SUBGHZ_LPLDLENR)

Address offset: 0x702

Reset value: 0x00

7	6	5	4	3	2	1	0
PLDLEN[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **PLDLEN[7:0]:** Payload length in number of bytes, in case of LoRa fixed header

4.10.30 Sub-GHz radio synchro timeout register (SUBGHZ_LSYNCTIMEOUTR)

Address offset: 0x706

Reset value: 0x00

7	6	5	4	3	2	1	0
SYNCTIMEOUT[7:0]							
r	r	r	r	r	r	r	r

Bits 7:0 **SYNCTIMEOUT[7:0]:** TimeoutValue = synchtimeout[7:3]*2^(2*synchtimeout[2:0]+1)
If a detection has not occurred by TimeoutValue, it goes back to Standby mode, or restart synch in continuous receive mode
Bits 7:3 synchtimeout(7:3) mantissa part
Bits 2:0 synchtimeout(2:0) exponent part

4.10.31 Sub-GHz radio Lora IQ polarity MSB register (SUBGHZ_LIQPOLR)

Address offset: 0x735

Reset value: 0x00

7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res

Bits 7:0 Reserved, must be kept at reset value.

4.10.32 Sub-GHz radio Lora IQ polarity LSB register (SUBGHZ_LIQPOLR)

Address offset: 0x736

Reset value: 0x00

7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res

Bits 7:0 Reserved, must be kept at reset value.

4.10.33 Sub-GHz radio LoRa synchronization word MSB register (SUBGHZ_LSYNCRH)

Address offset: 0x740

Reset value: 0x14

7	6	5	4	3	2	1	0
SYNCWORD[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[15:8]:** LoRa synchronization word MSB bits [15:8]
0x14: LoRa private network
0x34: LoRa public network
Others: reserved

4.10.34 Sub-GHz radio LoRa synchronization word LSB register (SUBGHZ_LSYNCRL)

Address offset: 0x741

Reset value: 0x24

7	6	5	4	3	2	1	0
SYNCWORD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SYNCWORD[7:0]:** LoRa synchronization word LSB bits [7:0]
0x24: LoRa private network
0x44: LoRa public network
Others: reserved

4.10.35 Sub-GHz radio Tx address pointer register (SUBGHZ_TXADRPTR)

Address offset: 0x0802

Reset value: 0x00

7	6	5	4	3	2	1	0
PTR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **PTR[7:0]:** TX buffer address pointer

4.10.36 Sub-GHz radio Rx address pointer register (SUBGHZ_RXADRPTRR)

Address offset: 0x0803

Reset value: 0x00

7	6	5	4	3	2	1	0
PTR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **PTR[7:0]:** RX buffer address pointer

4.10.37 Sub-GHz radio bandwidth select register (SUBGHZ_BWSELR)

Address offset: 0x807

Reset value: 0x00

7	6	5	4	3	2	1	0
Res	Res	Res	CHBWMANT[1:0]		CHBWEXPO[2:0]		
			r	r	r	r	r

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:3 **CHBWMANT[1:0]:** Channel bandwidth mantissa

Bits 2:0 **CHBWEXP[2:0]:** Channel bandwidth exponent

4.10.38 Sub-GHz radio random number register 3 (SUBGHZ_RNGR3)

Address offset: 0x819

Reset value: 0x00

7	6	5	4	3	2	1	0
RNDATA[31:24]							
r	r	r	r	r	r	r	r

Bits 7:0 **RNDATA[31:24]:** Random number data bits [31:24]

4.10.39 Sub-GHz radio random number register 2 (SUBGHZ_RNGR2)

Address offset: 0x81A

Reset value: 0x00

7	6	5	4	3	2	1	0
RNDATA[23:16]							
r	r	r	r	r	r	r	r

Bits 7:0 **RNDATA[23:16]:** Random number data bits [23:16]

4.10.40 Sub-GHz radio random number register 1 (SUBGHZ_RNGR1)

Address offset: 0x81B

Reset value: 0x00

7	6	5	4	3	2	1	0
RNDATA[15:8]							
r	r	r	r	r	r	r	r

Bits 7:0 **RNDATA[15:8]:** Random number data bits [15:8]

4.10.41 Sub-GHz radio random number register 0 (SUBGHZ_RNGR0)

Address offset: 0x81C

Reset value: 0x00

7	6	5	4	3	2	1	0
RNDATA[7:0]							
r	r	r	r	r	r	r	r

Bits 7:0 **RNDATA[7:0]:** Random number data bits [7:0]

4.10.42 Sub-GHz radio SD resolution register (SUBGHZ_SDCFG0R)

Address offset: 0x889

Reset value: 0x00

7	6	5	4	3	2	1	0
SD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SD[7:0]:** Radio SD resolution

4.10.43 Sub-GHz radio AGC RSSI control register (SUBGHZ_AGCRSSICTL0R)

Address offset: 0x89D

Reset value: 0x00

7	6	5	4	3	2	1	0
CALDATA[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CALDATA[7:0]:** AGC RSSI control

4.10.44 Sub-GHz radio receiver gain control register (SUBGHZ_RXGAINCR)

Address offset: 0x8AC

Reset value: 0x94

7	6	5	4	3	2	1	0
SENSI_ADJUST[5:0]						PMODE[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:2 **SENSI_ADJUST[5:0]:** Sensitivity Floor of AGC
This bitfield must be kept at 0x25.

Bits 1:0 **PMODE[1:0]:** Receiver power mode selection between normal mode and power saving mode
00: power saving mode (reduced sensitivity)
01: boost mode level1 active (improves sensitivity at detriment of power consumption)
10: boost mode level2 active (improves a set further sensitivity at detriment of power consumption)
Others: boost mode (best receiver sensitivity)

4.10.45 Sub-GHz radio AGC reset configuration register (SUBGHZ_AGCGFORSTCFGR)

Address offset: 0x8B8

Reset value: 0x14

7	6	5	4	3	2	1	0
EN[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **EN[7:0]:** Enable the reset generation for frequency-offset estimation

4.10.46 Sub-GHz radio AGC reset power threshold register (SUBGHZ_AGCGFORSTPOWTHR)

Address offset: 0x8B9

Reset value: 0x0A

7	6	5	4	3	2	1	0
PWRTHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **PWRTHR[7:0]:** Power threshold reset for frequency-offset estimation

4.10.47 Sub-GHz radio Tx clamp register (SUBGHZ_TXCLAMPR)

Address offset: 0x8D8

Reset value: 0x00

7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res

Bits 7:0 Reserved, must be kept at reset value.

4.10.48 Sub-GHz radio disable LNA register (REG_ANA_LNA)

Address offset: 0x8E2

Reset value: 0x00

7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res

Bits 7:0 Reserved, must be kept at reset value.

4.10.49 Sub-GHz radio disable mixer register (REG_ANA_MIXER)

Address offset: 0x8E5

Reset value: 0x00

7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res

Bits 7:0 Reserved, must be kept at reset value.

4.10.50 Sub-GHz radio PA over current protection register (SUBGHZ_PAOCPR)

Address offset: 0x8E7

Reset value: 0x18

7	6	5	4	3	2	1	0
Res.	Res.	OCP[5:0]					
		rw	rw	rw	rw	rw	rw

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **OCP[5:0]:** Power amplifier over current protection level
0x18: maximum 60 mA current for LP PA mode
0x38: maximum 140mA current for HP PA mode.
Others: reserved

4.10.51 Sub-GHz radio RTC control register (SUBGHZ_RTCCTLR)

Address offset: 0x902

Reset value: 0x00

7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	RTCEN
							rw

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **RTCEN:** Writing 1 restarts the radio RTC.

4.10.52 Sub-GHz radio RTC period MSB register (SUBGHZ_RTCPRDR2)

Address offset: 0x906

Reset value: 0x00

7	6	5	4	3	2	1	0
RTCPRD[31:16]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RTCPRD[31:16]:** Updates radio RTC period (MSB)

4.10.53 Sub-GHz radio RTC period mid-byte register (SUBGHZ_RTCPRDR1)

Address offset: 0x907

Reset value: 0x00

7	6	5	4	3	2	1	0
RTCPRD[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RTCPRD[15:8]:** Updates radio RTC period (mid-byte)

4.10.54 Sub-GHz radio RTC period LSB register (SUBGHZ_RTCPRDR0)

Address offset: 0x908

Reset value: 0x00

7	6	5	4	3	2	1	0
RTCPRD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RTCPRD[7:0]:** Updates radio RTC period (LSB)

4.10.55 Sub-GHz radio HSE32 OSC_IN capacitor trim register (SUBGHZ_HSEINTRIMR)

Address offset: 0x911

Reset value: 0x12

This register is retained in Sleep mode, but lost in Deep-Sleep mode.

7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]					
		rw	rw	rw	rw	rw	rw

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **TRIM[5:0]:** HSE32 XTAL mode OSC_IN load capacitor trimming
Load capacitor trimming step size ~0.47 pf.
0x00: minimum value ~11.3 pF
...
0x12 value ~20.3 pF (default)
...
0x2F: maximum capacitor value ~33.4 pF
Others: reserved

4.10.56 Sub-GHz radio HSE32 OSC_OUT capacitor trim register (SUBGHZ_HSEOUTTRIMR)

Address offset: 0x912

Reset value: 0x12

This register is retained in Sleep mode, but lost in Deep-Sleep mode.

7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]					
		rw	rw	rw	rw	rw	rw

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **TRIM[5:0]:** HSE32 XTAL mode OSC_OUT load capacitor trimming
Load capacitor trimming step size ~0.47 pf.
0x00: minimum value ~11.3 pF
...
0x12 value ~20.3 pF (default)
...
0x2F: maximum capacitor value ~33.4 pF
Others: reserved

4.10.57 Sub-GHz radio SMPS control 0 register (SUBGHZ_SMPSC0R)

Address offset: 0x916

Reset value: 0x00

7	6	5	4	3	2	1	0
Res.	CLKDE	Res.	Res.	Res.	Res.	Res.	Res.
	rw						

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CLKDE:** SMPS clock detection enable

SMPS clock detection must be enabled before enabling the SMPS, if the application uses an external HSE clock source (not coming from XO or TCXO but from another device).

0: SMPS clock detection disabled

1: SMPS clock detection enabled

Bits 5:0 Reserved, must be kept at reset value.

4.10.58 Sub-GHz radio power control register (SUBGHZ_PCR)

Address offset: 0x91A

Reset value: 0x50

This register is retained in Sleep mode but lost in Deep-Sleep mode.

7	6	5	4	3	2	1	0
Res.	CLE	CLV[1:0]		Res.	Res.	Res.	Res.
	rw	rw	rw				

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CLE:** Power-supply current limiter enable

0: power-supply current limiter disabled (unlimited current)

1: power-supply current limiter enabled (current limited according to CLV[1:0])

Bits 5:4 **CLV[1:0]:** Power-supply current limiter value

When the power-supply current limiter is enabled by CLEN, these bits define the maximum current limiting level.

0x0: power-supply current limiting level 25 mA

0x1: power-supply current limiting level 50 mA (default)

0x2: power-supply current limiting level 100 mA

0x3: power-supply current limiting level 200 mA

Bits 3:0 Reserved, must be kept at reset value.

4.10.59 Sub-GHz radio SMPS control 2 register (SUBGHZ_SMPSC2R)

Address offset: 0x923

Reset value: 0x06

This register is retained in Sleep mode but lost in Deep-Sleep mode.

7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	DRV[1:0]		Res
					rw	rw	

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:1 **DRV[1:0]:** SMPS maximum drive capability.
0x0: 20 mA
0x1: 40 mA
0x2: 60 mA
0x3: 100 mA (default)

Bit 0 Reserved, must be kept at reset value.

4.10.60 Sub-GHz radio RTC control register (SUBGHZ_EVENTMASKR)

Address offset: 0x944

Reset value: 0x00

7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res

Bits 7:0 Reserved, must be kept at reset value.

4.10.61 Sub-GHz radio register map

Table 34. SUBGHZ register map and reset values

Offset	Register name	7	6	5	4	3	2	1	0
0x6AC	SUBGHZ_GBSYNCR	Res.	SBITSYNCEN	RXDINV	BITSYNCDIS	Res.	Res.	Res.	Res.
	Reset value		0	0	0				
0x6AC-0x6B4	Reserved	Reserved							
0x6B8	SUBGHZ_GPKTCTL1AR	Res.	Res.	SYNCDETEN	CONTTX	INFSQEQSEL[1:0]		INFSSQEQEN	WHITEINI[8]
	Reset value			1	0	0	0	0	1
0x6B9	SUBGHZ_GWHITEINIRL	WHITEINI[7:0]							
	Reset value	0	0	0	0	0	0	0	0
0x6BC	SUBGHZ_GCRCINIRH	CRCINI[15:8]							
	Reset value	0	0	0	1	1	1	0	1
0x6BD	SUBGHZ_GCRCINIRL	CRCINI[7:0]							
	Reset value	0	0	0	0	1	1	1	1
0x6BE	SUBGHZ_GCRCPOLRH	CRCPOL[15:8]							
	Reset value	0	0	0	1	0	0	0	0

Table 34. SUBGHZ register map and reset values (continued)

Offset	Register name	7	6	5	4	3	2	1	0
0x6BF	SUBGHZ_GCRCPOLRL	CRCPOL[7:0]							
	Reset value	0	0	1	0	0	0	0	1
0x6C0	SUBGHZ_GSYNCR7	SYNCWORD[63:56]							
	Reset value	1	0	0	1	0	1	1	1
0x6C1	SUBGHZ_GSYNCR6	SYNCWORD[55:48]							
	Reset value	0	0	1	0	0	0	1	1
0x6C2	SUBGHZ_GSYNCR5	SYNCWORD[47:40]							
	Reset value	0	1	0	1	0	0	1	0
0x6C3	SUBGHZ_GSYNCR4	SYNCWORD[39:32]							
	Reset value	0	0	1	0	0	1	0	1
0x6C4	SUBGHZ_GSYNCR3	SYNCWORD[31:24]							
	Reset value	0	1	0	1	0	1	1	0
0x6C5	SUBGHZ_GSYNCR2	SYNCWORD[23:16]							
	Reset value	0	1	0	1	0	0	1	1
0x6C6	SUBGHZ_GSYNCR1	SYNCWORD[15:8]							
	Reset value	0	1	1	0	0	1	0	1
0x6C7	SUBGHZ_GSYNCR0	SYNCWORD[7:0]							
	Reset value	0	1	1	0	0	1	0	0
0x6C8-0x73C	Reserved	Reserved							
0x740	SUBGHZ_LSYNCRH	SYNCWORD[15:8]							
	Reset value	0	0	0	1	0	1	0	0
0x741	SUBGHZ_LSYNCRL	SYNCWORD[7:0]							
	Reset value	0	0	1	0	0	1	0	0
0x742-0x818	Reserved	Reserved							
0x819	SUBGHZ_RNGR3	RNDATA[31:24]							
	Reset value	0	0	0	0	0	0	0	0
0x81A	SUBGHZ_RNGR2	RNDATA[23:16]							
	Reset value	0	0	0	0	0	0	0	0
0x81B	SUBGHZ_RNGR1	RNDATA[15:8]							
	Reset value	0	0	0	0	0	0	0	0
0x81C	SUBGHZ_RNGR0	RNDATA[7:0]							
	Reset value	0	0	0	0	0	0	0	0
0x820-0x8AB	Reserved	Reserved							
0x8AC	SUBGHZ_RXGAINCR	SENSI_ADJUST[5:0].						PMODE[1:0]	
	Reset value	1	0	0	1	0	1	0	0
0x8B0-0x8E6	Reserved	Reserved							
0x8E7	SUBGHZ_PAOCPR	Res.	Res.	OCP[5:0]					
	Reset value			0	1	1	0	0	0
0x8E8 - 0x910	Reserved	Reserved							
0x911	SUBGHZ_HSEINTRIMR	Res.	Res.	TRIM[5:0]					
	Reset value			0	1	0	0	1	0
0x912	SUBGHZ_HSEOUTTRIMR	Res.	Res.	TRIM[5:0]					
	Reset value			0	1	0	0	1	0

Table 34. SUBGHZ register map and reset values (continued)

Offset	Register name	7	6	5	4	3	2	1	0
0x913-0x915	Reserved	Reserved							
0x916	SUBGHZ_SMPSC0R	Res.	CLKDE	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0						
0x917-0x919	Reserved	Reserved							
0x91A	SUBGHZ_PCR	Res.	CLE	CLV[1:0]		Res.	Res.	Res.	Res.
	Reset value		1	0	1				
0x91B - 0x920	Reserved	Reserved							
0x923	SUBGHZ_SMPSC2R	Res.	Res.	Res.	Res.	Res.	DRV[1:0]		Res.
	Reset value						1	1	

Refer to *Section 2.4* for the register boundary addresses.

5 Power control (PWR)

5.1 Power supplies

The STM32WLEx devices require a V_{DD} operating voltage supply between 1.71 V and 3.6 V. Several independent supplies (V_{DDSMPS}, V_{FBSMPS}, V_{DDA}, V_{DDRF}) can be provided for specific peripherals:

- V_{DD} = 1.71 V to 3.6 V

  V_{DD} is the external power supply for the I/Os, the system analog blocks such as reset, power management, internal clocks and low-power regulator. It is provided externally through VDD pins.

- V_{DDSMPS} = 1.71 V to 3.6 V

  V_{DDSMPS} is the external power supply for the SMPS step-down converter. It is provided externally through VDDSMPS supply pin and must be connected to the same supply as V_{DD}.

- V_{FBSMPS} = 1.55 V

  V_{FBSMPS} is the external power supply for the main system regulator. It is provided externally through VFBSMPS pin and is supplied through the SMPS step-down converter.

- V_{DDA} = 0 V to 3.6 V (DAC/COMPs minimum voltage is 1.62 V, ADC minimum voltage is 1.8 V and VREFBUF minimum voltage is 2.4 V).

  V_{DDA} is the external analog power supply for A/D converters, D/A converters, voltage reference buffer, and comparators. The V_{DDA} voltage level is independent from the V_{DD} voltage (see power-up and power-down limitations below) and must preferably be connected to V_{DD} when these peripherals are not used.

- V_{DDRF} = 1.71 V to 3.6 V

  V_{DDRF} is the external power supply for the radio. It is provided externally through the VDDRF pin and must be connected to the same supply as V_{DD}.

- $V_{DDRF1V5}$ = 1.45 V to 1.62 V

  $V_{DDRF1V5}$ is the external power supply for the radio. It is provided externally through the VDDRF1V5 pin.

- V_{BAT} = 1.55 V to 3.6 V

  V_{BAT} is the power supply for RTC, TAMP, external clock 32 kHz oscillator and backup registers (through power switch) when V_{DD} is not present.

- V_{REF-}, V_{REF+}

  V_{REF+} is the input reference voltage for ADC. It is also the output of the internal voltage reference buffer when enabled.

  – When V_{DDA} < 2 V, V_{REF+} must be equal to V_{DDA}.

  – When V_{DDA} ≥ 2 V, V_{REF+} must be between 2 V and V_{DDA}.

  V_{REF+} can be grounded when ADC is not active. The internal voltage reference buffer supports the following output voltages, configured with VRS bit in the VREFBUF_CSR register:

  – V_{REF+} around 2.048 V: this requires V_{DDA} ≥ 2.4 V.

  – V_{REF+} around 2.5 V: this requires V_{DDA} ≥ 2.8 V.

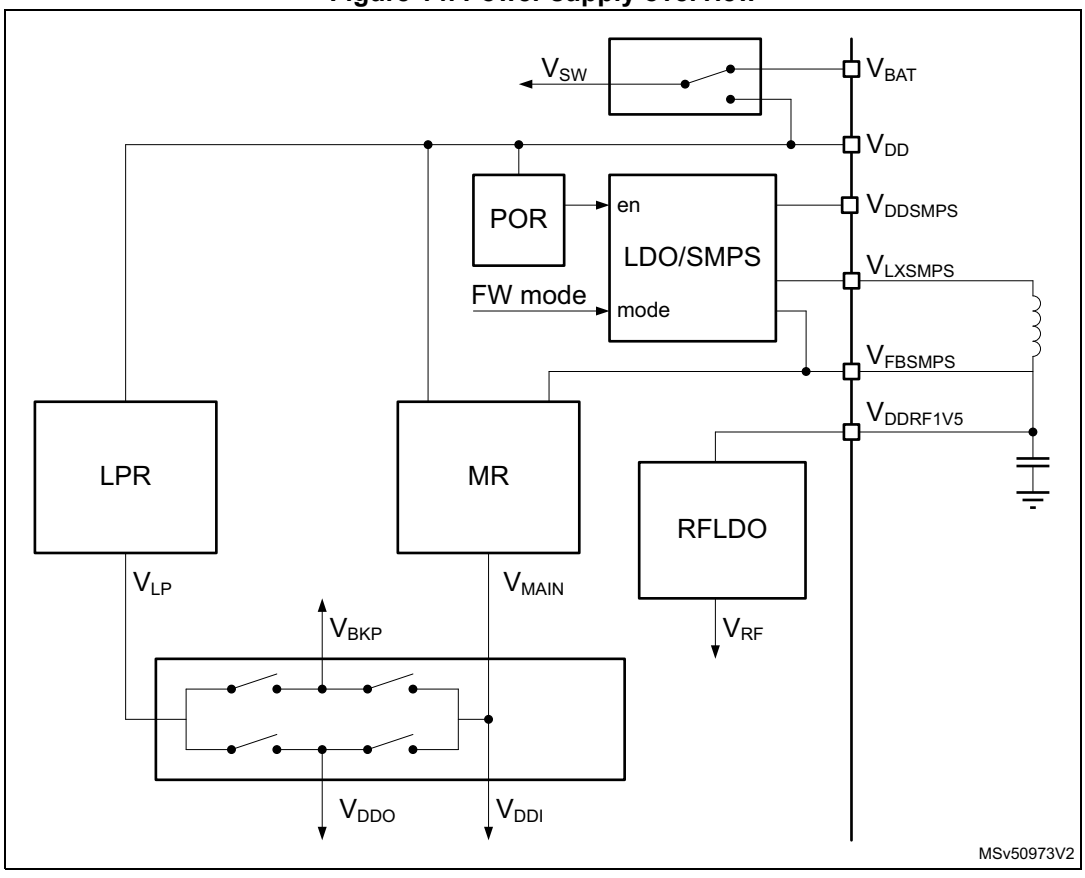
VREF+ pin is not available on all packages. When not available, this pin is internally bonded to VDDA. When VREF+ is double-bonded with VDDA in a package, the internal voltage reference buffer is not available and must be kept disabled (refer to the datasheet for pinout descriptions).

During power up and power down, the following power sequence is required:

1. When V_{DD} < 1 V other power supplies (V_{DDA}) must remain below V_{DD} + 300 mV.
   During power down, V_{DD} can temporarily become lower then other supplies only if the energy provided to the device remains below 1 mJ. This allows external decoupling capacitors to be discharged with different time constants during this transient phase.
2. When V_{DD} > 1 V, all other power supplies (V_{DDA}) become independent.

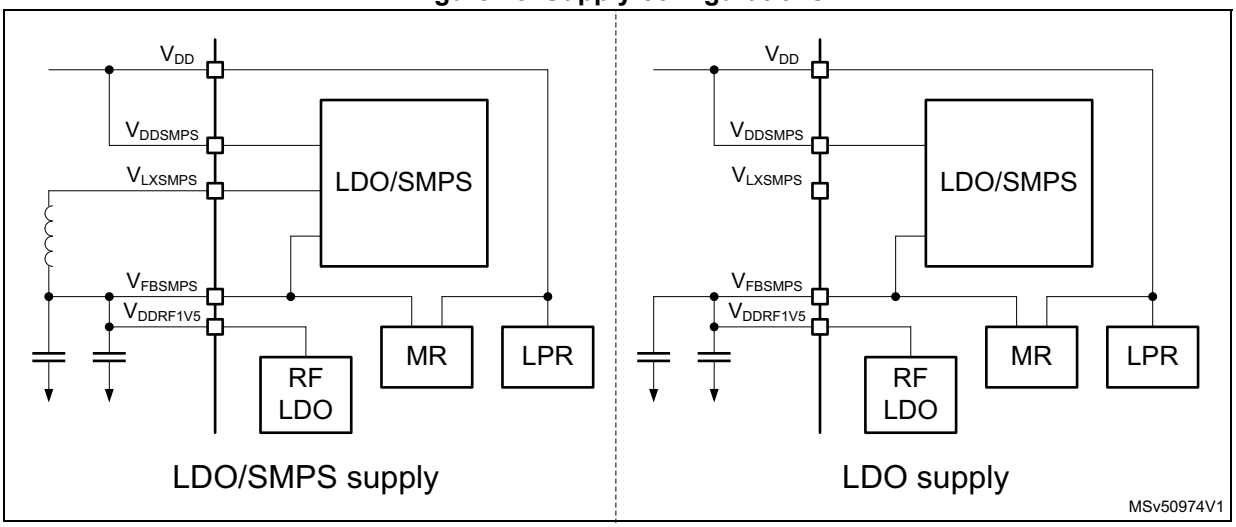
An embedded linear voltage regulator is used to supply the internal digital power V_{CORE}. V_{CORE} is the power supply for digital peripherals, SRAM1 and SRAM2. The flash memory is supplied by V_{CORE} and V_{DD}. V_{CORE} is split in two parts: V_{DDO} part and an interruptible part V_{DDI}.

Figure 14. Power supply overview



The different supply configurations are shown in the figure below.

Figure 15. Supply configurations



The LDO or SMPS step-down converter operating mode can be configured by one of the following:

- by the MCU using the SMPSEN setting in *PWR control register 5 (PWR_CR5)*, that depends upon the MCU system operating mode (Run, Stop, Standby or Shutdown).
- by the sub-GHz radio using `Set_RegulatorMode()` command and the sub-GHz radio operating mode (Sleep, Calibrate, Standby, Standby with HSE32 or Active).

After any POR and NRST reset, the LDO mode is selected. The SMPS selection has priority over LDO selection.

While the sub-GHz radio is in Standby with HSE32 or in Active mode, the supply mode is not altered until the sub-GHz radio enters Standby or Sleep mode. The sub-GHz radio activity may add a delay for entering the MCU software requested supply mode.

The LDO or SMPS supply mode can be checked with the SMPSRDY flag in *Power status register 2 (PWR_SR2)*.

Note: When the radio is active, the supply mode is not changed until after the radio activity is finished.

During Stop 1, Stop 2 and Standby modes, when the sub-GHz radio is not active, the LDO or SMPS step-down converter is switched off. When exiting low-power modes (except Shutdown), the SMPS step-down converter is set by hardware to the mode selected by the SMPSEN bit in *PWR control register 5 (PWR_CR5)*. SMPSEN is retained in Stop and Standby modes.

Independently from the MCU software selected supply operating mode, the sub-GHz radio allows the supply mode selection while the sub-GHz radio is active (thanks to the sub-GHz radio `Set_RegulatorMode()`command*)*. For more details, see *Relation between MCU and sub-GHz radio operating modes*.

The maximum load current delivered by the SMPS can be selected by the sub- GHz radio SUBGHZ_SMPSC2R register. For more details see *Section 4: Sub-GHz radio (SUBGHZ)*.

The inrush current of the LDO and SMPS step-down converter can be controlled via the sub- GHz radio SUBGHZ_PCR register. This information is retained in all but the sub-GHz radio Deep-Sleep mode. For more details see *Section 4: Sub-GHz radio (SUBGHZ)*.

The SMPS needs a clock to be functional. If for any reason this clock stops, the device may be destroyed. It can be the case if the HSE is provided by an external clock source (*Figure 22: HSE32 clock sources*), with the risk that this clock disappears while the SMPS is enabled. To avoid this situation, a clock detection is used to, in case of a clock failure, switch off the SMPS and enable the LDO. The SMPS clock detection is enabled by the sub-GHz radio SUBGHZ_SMPSC0R.CLKDE. By default, the SMPS clock detection is disabled and must be enabled before enabling the SMPS. For more details, see *Section 4: Sub-GHz radio (SUBGHZ)*.

5.1.1 Independent analog peripherals supply

To improve the ADC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply that can be separately filtered and shielded from noise on the PCB.

The analog peripherals voltage supply input is available on a separate VDDA pin.

An isolated supply ground connection is provided on VSSA pin.

The V_{DDA} supply voltage can be different from V_{DD}. The presence of V_{DDA} must be checked before enabling any of the analog peripherals supplied by V_{DDA} (A/D converter, comparators, voltage reference buffer).

The V_{DDA} supply can be monitored by the peripheral voltage monitoring, and compared with a threshold (1.65 V for PVM3). See *Section 5.2.3: Peripheral voltage monitoring (PVM)* for more details.

When a single supply is used, V_{DDA} can be externally connected to V_{DD} through the external filtering circuit in order to ensure a noise-free V_{DDA} reference voltage.

ADC reference voltage

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect a separate reference voltage lower than V_{DDA}, to V_{REF+}. V_{REF+} is the highest voltage, represented by the full scale value, for an analog input (ADC) signal.

V_{REF+} can be provided either by an external reference of by an internal buffered voltage reference (VREFBUF).

The internal voltage reference is enabled by setting the ENVR bit in the *VREFBUF control and status register (VREFBUF_CSR)*. The voltage reference is set to 2.5 V when the VRS bit is set and to 2.048 V when the VRS bit is cleared. The internal voltage reference can also provide the voltage to external components through V_{REF+} pin. Refer to the device datasheet and to *Voltage reference buffer (VREFBUF)* for further information.

5.1.2 Battery Backup domain

To retain the content of the backup registers and supply the RTC and TAMP functions when V_{DD} is turned off, the VBAT pin can be connected to an optional backup voltage supplied by a battery or by another source.

The VBAT pin powers RTC, TAMP, the LSE oscillator and the PC13 to PC15 I/Os, allowing RTC and TAMP to operate even when the main power supply is turned off. The switch to the V_{BAT} supply is controlled by the power-down reset embedded in the reset block.

Warning: **During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.**
During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (refer to the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current can be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

If no external battery is used in the application, it is recommended to connect VBAT externally to V_{DD} with a 100 nF external ceramic decoupling capacitor.

When the Backup domain is supplied by V_{DD} (analog switch connected to V_{DD}), the following pins are available:

- PC13, PC14 and PC15, that can be used as GPIO pins
- PC13, PC14 and PC15, that can be configured by RTC, TAMP or LSE (refer to *Section 30: Real-time clock (RTC)* and *Section 31: Tamper and backup registers (TAMP)*)
- PA0/TAMP_IN2 and PB3/TAMP_IN3 when they are configured by the TAMP as tamper pins

Note: Due to the fact that the analog switch can transfer only a limited amount of current (3 mA), the use of GPIO PC13 to PC15 in output mode is restricted: the speed must be limited to 2 MHz with a maximum load of 30 pF and these I/Os must not be used as current source (e.g. to drive a LED).

When the Backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT} because V_{DD} is not present), the following functions are available:

- PC13, PC14 and PC15 can be controlled only by RTC, TAMP or LSE (refer to *Section 30: Real-time clock (RTC)* and *Section 31: Tamper and backup registers (TAMP)*)
- PA0/TAMP_IN2 and PB3/TAMP_IN3, when they are configured by the TAMP as tamper pins

Backup domain access

After a system reset, the Backup domain (RTC and TAMP backup registers) is protected against possible unwanted write accesses. The DBP bit must be set in the *PWR control register 1 (PWR_CR1)* to enable access to the Backup domain

VBAT battery charging

When V_{DD} is present, It is possible to charge the external battery on VBAT through an internal resistance.

The VBAT charging is done either through a 5 kΩ resistor or through a 1.5 kΩ resistor, depending on the VBRS bit value in the *PWR control register 4 (PWR_CR4)*.

The battery charging is enabled by setting VBE bit in the *PWR control register 4 (PWR_CR4)*, and automatically disabled in VBAT mode.

5.1.3 Voltage regulator

Two embedded linear voltage regulators supply all the digital circuitries, except for the Standby circuitry and the Backup domain. The main regulator (MR) output voltage (V_{CORE}) can be programmed by software to two different power ranges (range 1 and range 2) to optimize the consumption depending on the system maximum operating frequency (refer to *Section 6.2.9: Clock source frequency versus voltage scaling* and to *Section 3.3.4: Read access latency*.

The voltage regulators are always enabled after a reset. Depending on the application modes, the V_{CORE} supply is provided either by the main regulator or by the low-power regulator (LPR), as detailed below:

- In Run, Sleep and Stop 0 modes, both regulators are enabled and the main regulator (MR) supplies full power to the V_{CORE} domain (core, memories and digital peripherals).
- In LPRun and LPSleep modes, the main regulator (MR) is off and the low-power regulator (LPR) supplies reduced power to the V_{CORE} domain, preserving the contents of the registers and internal SRAM1 and SRAM2.
- In Stop 1 and Stop 2 modes, the main regulator (MR) is off and the low-power regulator (LPR) supplies low power to the all or part of the V_{CORE} domain, preserving the contents of all or part of the registers and of internal SRAM1 and SRAM2.
- In Standby modes with SRAM2 content preserved (RRS bit set in the *PWR control register 3 (PWR_CR3)*), the main regulator (MR) is off and the low-power regulator (LPR) provides the supply to SRAM2 only. The core and digital peripherals (except Standby circuitry and Backup domain), SRAM1 is powered off.
- In Standby mode, both regulators (MR and LPR) are powered off. The contents of the registers and of SRAM1 and SRAM2 is lost except for the Standby circuitry and the Backup domain.
- In Shutdown mode, both regulators are powered off. When exiting from Shutdown mode, a power-on reset is generated. Consequently, the contents of the registers and of both SRAM1 and SRAM2 is lost, except for the Backup domain.

5.1.4 Dynamic voltage scaling management

The dynamic voltage scaling is a power management technique that consists in increasing or decreasing the voltage used for the digital peripherals (V_{CORE}), according to the application performance and power consumption needs.

Dynamic voltage scaling to increase V_{CORE} is known as “overvolting”. It is used to improve the device performance.

Dynamic voltage scaling to decrease V_{CORE} is known as “undervolting”. It is used to save power, particularly in laptop and other mobile devices where the energy comes from a battery and is thus limited.

- range 1: high-performance range

  The main regulator provides a typical output voltage at 1.2 V. The system clock frequency can be up to 64 MHz. The flash memory access time for read access is minimum. Write and erase operations are possible.
- range 2: low-power range

  The main regulator provides a typical output voltage at 1.0 V. The system clock frequency can be up to 16 MHz.The flash memory access time for a read access is increased as compared to range 1. Write and erase operations are possible.

Voltage scaling is selected through the VOS bit in the *PWR control register 1 (PWR_CR1)*.

The sequence to go from range 1 to range 2 is the following:

1. Reduce the system frequency to a value lower or equal to 16 MHz.
2. Adjust number of wait states according to new frequency target in range 2 (LATENCY bits in the FLASH_ACR).
3. Select range 2 with the VOS bits in the *PWR control register 1 (PWR_CR1)*.

The sequence to go from range 2 to range 1 is detailed below:

1. Select range 1 in the VOS bits in the *PWR control register 1 (PWR_CR1)*.
2. Wait until the VOSF flag is cleared in the *Power status register 2 (PWR_SR2)*.
3. Adjust number of wait states according new frequency target in range 1 (LATENCY bits in the FLASH_ACR).
4. Increase the system frequency.

5.2 Power supply supervisor

5.2.1 Power-on reset (POR)/power-down reset (PDR) /Brownout reset (BOR)

The device has an integrated power-on reset/power-down reset, coupled with a Brownout reset circuitry.

Five BOR thresholds can be selected through option bytes.

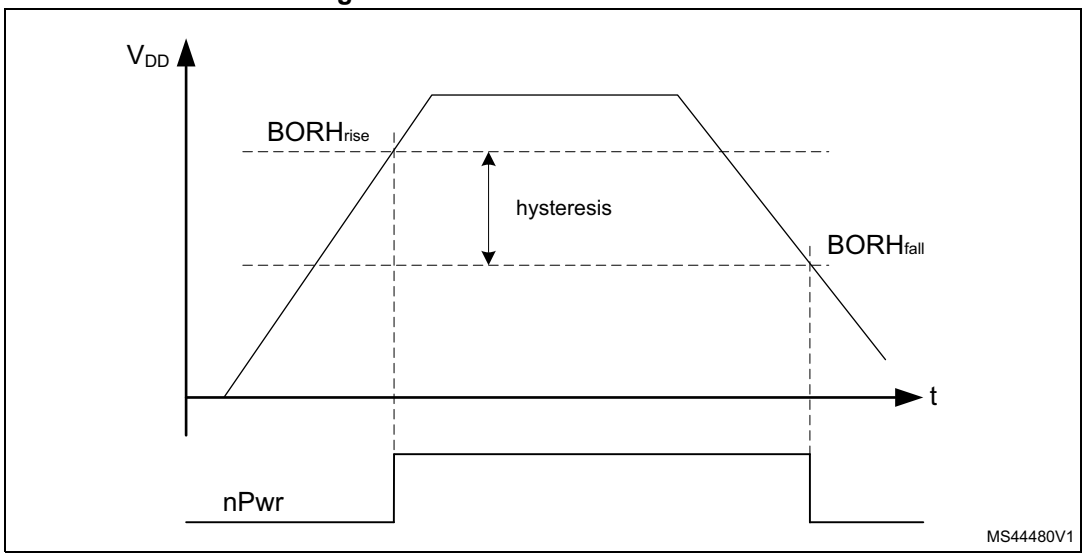
BOR0 level cannot be disabled. Other BOR levels can be enabled by user option. When enabled, BOR is active in all power modes except in Shutdown.

Reset mode

During power-on, BOR keeps the device under reset until the supply voltage V_{DD} reaches the specified V_{BORx} threshold. When V_{DD} drops below the selected threshold, a device reset is generated. When V_{DD} is above the V_{BORx} upper limit, the device reset is released and the system can start.

For more details on the Brownout reset thresholds, refer to the electrical characteristics section in the datasheet.

Figure 16. Brownout reset waveform



1. The reset temporization $t_{RSTTEMPO}$ is present only for the BOR lowest threshold (V_{BOR0}).

5.2.2 Programmable voltage detector (PVD)

The PVD can be used to monitor V_{DD} by comparing it to a threshold selected by the PLS[2:0] bits in the *PWR control register 2 (PWR_CR2)*.

PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case the voltage level on PVD_IN is compared to the internal VREFINT level.

PVD is enabled by setting the PVDE bit.

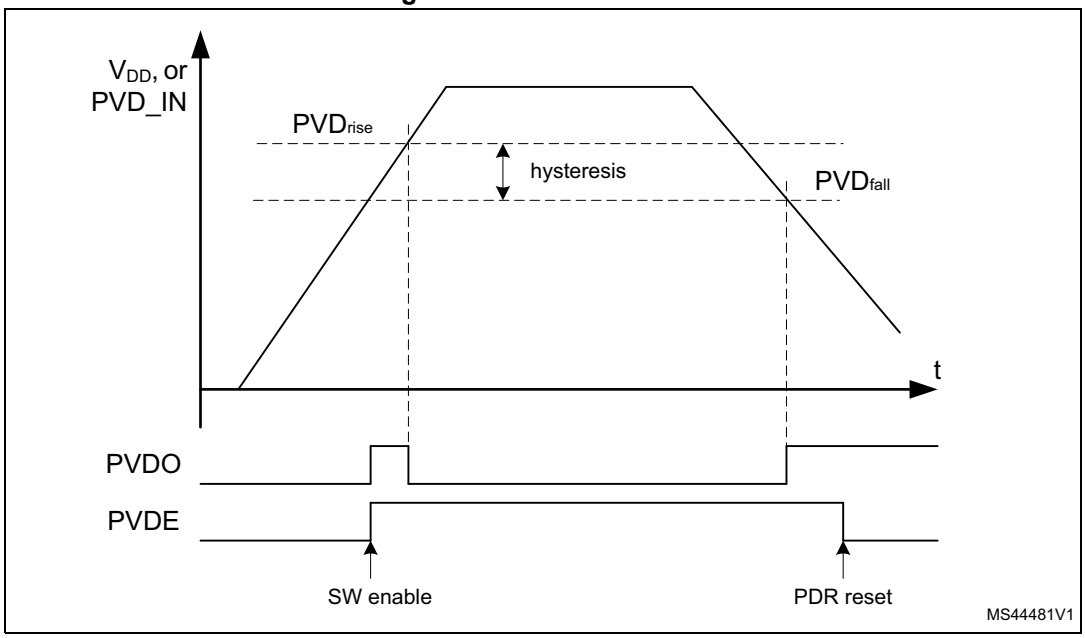
A PVDO flag is available in the *Power status register 2 (PWR_SR2)* to indicate if V_{DD} or the voltage level on PVD_IN is higher or lower than the PVD threshold. This event is internally connected to the EXTI line 16 and can generate an interrupt if enabled through the EXTI registers.

The PVD output interrupt can be generated when V_{DD} or voltage level on PVD_IN drops below the PVD threshold and/or when V_{DD} or voltage level on PVD_IN rises above the PVD threshold depending on EXTI line 16 rising/falling edge configuration. As an example, the service routine can perform emergency shutdown tasks.

The PVD can be configured to monitor the V_{DD} supply level needed for the sub-GHz radio operation. For this, the PVD must select its lowest level and PVD and the wakeup must be enabled in EWPVD. Only a voltage drop below the PVD level generates a wakeup event.

BOR0 level cannot be disabled. The other BOR levels can be enabled by user option. When enabled, BOR is active in all power modes except in Shutdown.

Figure 17. PVD thresholds



5.2.3 Peripheral voltage monitoring (PVM)

Only V_{DD} is monitored by default as it is the only supply required for all system-related functions. The other supplies (such as V_{DDA}) can be independent from V_{DD} and can be monitored by the peripheral voltage monitoring (PVM).

Each PVMx is a comparator between a fixed threshold V_{PVMx} and the selected power supply. PVMOx flags indicate if the independent power supply is higher or lower than the PVMx threshold. The PVMOx flag is cleared when the supply voltage is above the PVMx threshold and is set when the supply voltage is below the PVMx threshold.

Each PVM output is connected to an EXTI line and can generate an interrupt if enabled through the EXTI registers. The PVMx output interrupt is generated when the independent power supply drops below the PVMx threshold and/or when it rises above the PVMx threshold, depending on EXTI line rising/falling edge configuration.

Each PVM can remain active in Stop 0, Stop 1 and Stop 2 modes, and the PVM interrupt can wake up from the Stop modes.

Table 35. PVM features

PVM	Power supply	PVM threshold	EXTI line
PVM1	Not used	-	-
PVM2	Not used	-	-
PVM3	V_{DDA}	V_{PVM3} (around 1.65 V)	34
PVM4	Not used	-	-

The independent supply V_{DDA} is not considered as present by default and a logical and electrical isolation is applied to ignore any information coming from the peripherals supplied by these dedicated supplies:

- If V_{DDA} is shorted externally to V_{DD}, the application must assume that V_{DDA} is available without enabling any peripheral voltage monitoring.
- If V_{DDA} is independent from V_{DD}, the peripheral voltage monitoring (PVM) can be enabled to confirm whether the V_{DDA} supply is present or not.

The following sequence must be applied before using any of these analog peripherals: ADC, DAC, comparators or voltage reference buffer:

1. If V_{DDA} is independent from V_{DD}:
   a) Enable PVM3 by setting PVME3 bit in the *PWR control register 2 (PWR_CR2)*.
   b) Wait for the PVM3 wakeup time.
   c) Wait until PVMO3 is cleared in the *Power status register 2 (PWR_SR2)*.
   d) Disable the PVM3 for consumption saving (optional).
2. Enable the analog peripheral. This automatically removes the V_{DDA} isolation.

5.2.4 Radio end of life (EOL)

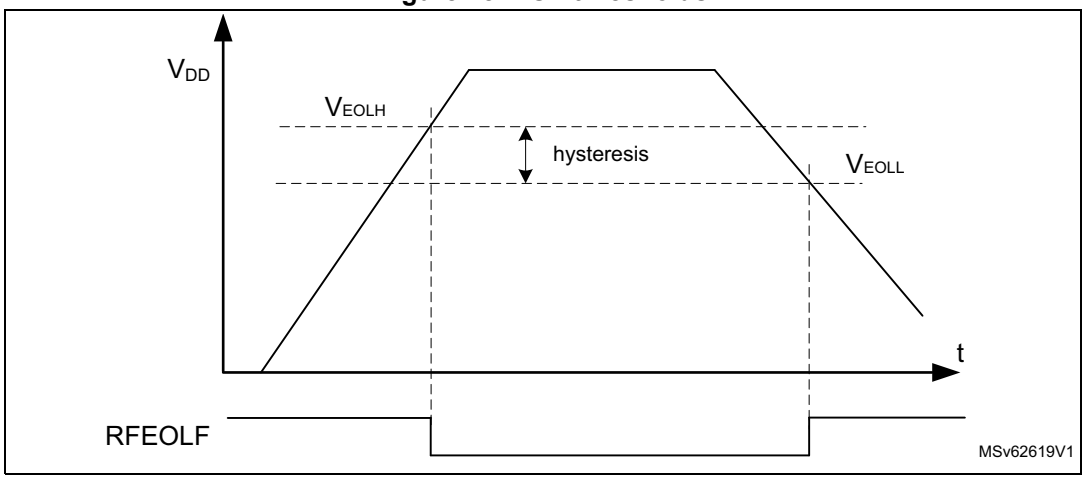
The radio end-of-life monitor provides information on the V_{DD} supply when it is too low to operate the sub-GHz radio. When reaching the EOL level, the software must stop all radio activity in a safe way.

The EOL is enabled by setting the RFEOLEN bit.

A RFEOLF flag is available in *Power status register 2 (PWR_SR2)* to indicate if V_{DD} voltage level is below the EOL threshold.

The EOL is only generated when the sub-GHz radio is in Calibrate, Standby or Active mode.

Figure 18. EOL thresholds

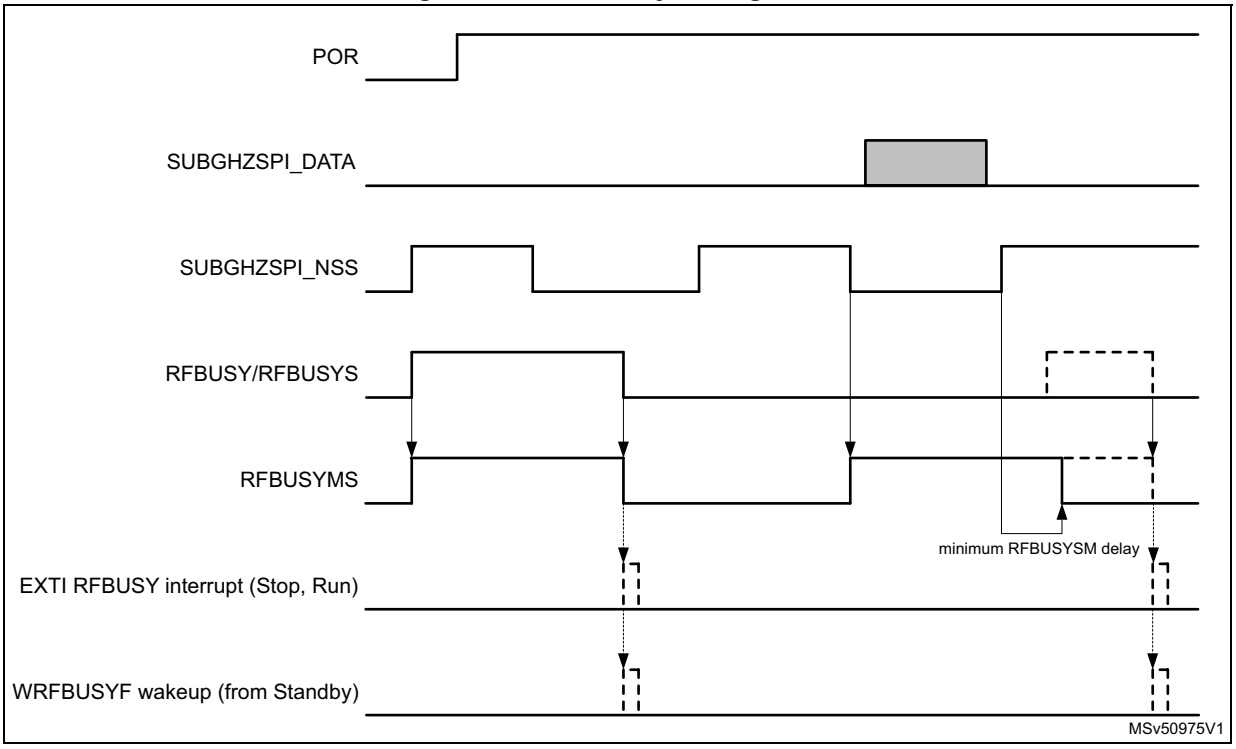


5.3 Radio busy management

For correct software handling of the radio busy signal RFBUSY, additional busy control are included in the PWR controller. This generates a busy mask RFBUSYMS status based on

the SUBGHZSPI_NSS activity, and masks the RFBUSYS status low time (not busy) after an SPI command transfer (see the figure below).

Figure 19. Radio busy management



At reset, the radio is busy (as signaled by the RFBUSY signal). At this time, the RFBUSYMS signal provides the same information. Subsequently the radio is woken up by a falling edge on SUBGHZSPI_NSS. Once the radio is ready to receive a command, RFBUSYS and RFBUSYMS go low. After reset, the RFBUSYS or RFBUSYMS status can be used to check that the radio is ready.

For any subsequent command, as soon as the SUBGHZSPI_NSS goes low, the RFBUSYMS status is set to signal busy radio. RFBUSYMS remains high for at least the minimum RFBUSYMS delay after the SUBGHZSPI_NSS is set high, or as long as the RFBUSY signal remains high:

- When transferring a command with an expected radio busy indication the RFBUSYMS status must be used to detect the radio busy state.
- When transferring a command without any expected radio busy indication, there is no need for checking any busy status. Anew command can be sent even when RFBUSYMS status indicates busy radio.

When the SUBGHZSPI_NSS is selected to be driven by the LPTIM3_OUT in PWR_CR1.SUBGHZSPINSSSEL, the RFBYSYMS status is disabled and must not be used by software.

When in Standby mode, the CPU can be woken up through a WRFBUSYF wakeup flag when enabled by EWRFBUSY in *Section 5.5.3: PWR control register 3 (PWR_CR3)*.

When in Stop or Run mode, a CPU can be woken up and interrupted by the EXTI configurable event on the RFBUSY signal, see *Section 14.3.1: EXTI wakeup interrupt list*.

5.4 Low-power modes

By default, the microcontroller is in Run mode after a system or a power reset. Low-power modes are available to save power when the CPU does not need to be kept running, for example when it is waiting for an external event. The user must select the mode giving the best compromise between consumption, startup time and available wakeup sources.

These low-power modes are detailed below:

- *Sleep mode*: CPU clock off, all peripherals including CPU core peripherals (among them NVIC, SysTick) can run and wake up the CPU when an interrupt or an event occurs.
- *Low-power run mode (LPRun)*: when the system clock frequency is reduced below 2 MHz. The code is executed from the SRAM or from the flash memory. The regulator is in low-power mode to minimize the operating current.
- *Low-power sleep mode (LPSleep)*: entered from the LPRun mode.
- *Stop 0 mode*, and *Stop 1 mode*: the content of SRAM1, SRAM2 and of all registers is retained. All clocks in the V_{CORE} domain are stopped. PLL, MSI, HSI16 and HSE32 are disabled. LSI and LSE can be kept running.

  RTC can remain active (Stop mode with RTC, Stop mode without RTC). The sub-GHz radio may remain active independently from the CPU.

  Some peripherals with the wakeup capability can enable HSI16 RC during the Stop mode to detect their wakeup condition.

  Stop 1 offers the largest number of active peripherals and wakeup sources, a smaller wakeup time but a higher consumption compared with Stop 2.

  In Stop 0 mode, the main regulator remains on, resulting in the fastest wakeup time but with much higher consumption. The active peripherals and wakeup sources are the same as in Stop 1 mode that uses the low-power regulator.

  The system clock, when exiting Stop 0 or Stop 1 mode, can be either MSI up to 48 MHz or HSI16, depending on the software configuration.
- *Stop 2 mode*: part of the V_{CORE} domain is powered off. Only SRAM1, SRAM2, CPU and some peripherals preserve their contents (see *Table 37: Functionalities depending on system operating mode*).

  All clocks in the V_{CORE} domain are stopped. PLL, MSI, HSI16 and HSE32 are disabled. LSI and LSE can be kept running.

  RTC can remain active (Stop 2 mode with RTC, Stop 2 mode without RTC). The sub-GHz radio may also remain active independent from the CPU.

  Some peripherals with the wakeup capability can enable HSI16 RC during the Stop 2 mode to detect their wakeup condition (see *Table 37: Functionalities depending on system operating mode*) .

  The system clock when exiting from Stop 2 mode, can be either MSI up to 48 MHz or HSI16, depending on the software configuration.
- *Standby mode*: V_{CORE} domain is powered off. However, it is possible to preserve the SRAM2 content as detailed below:
  - Standby mode with SRAM2 retention when the RRS bit is set in the *PWR control register 3 (PWR_CR3)*. In this case, SRAM2 is supplied by the low-power regulator.
  - Standby mode when the RRS bit is cleared in the *PWR control register 3 (PWR_CR3)*. In this case the main regulator and the low-power regulator are

powered off.

All clocks in the V_{CORE} domain are stopped. PLL, MSI, HSI16 and HSE32 are disabled. LSI and LSE can be kept running.

Th RTC can remain active (Standby mode with RTC, Standby mode without RTC). The sub-GHz radio and the PVD may also remain active when enabled independent from the CPU. In Standby mode, the PVD selects its lowest level.

The system clock, when exiting Standby modes, is MSI at 4 MHz.

- *Shutdown mode*: V_{CORE} domain is powered off. All clocks in the V_{CORE} domain are stopped. PLL, MSI, HSI16, LSI and HSE32 are disabled. LSE can be kept running. The system clock when exiting the Shutdown mode, is MSI at 4 MHz. In this mode, the supply voltage monitoring is disabled and the product behavior is not guaranteed in case of a power voltage drop.

In addition, the power consumption in Run mode can be reduced by slowing down the system clocks, and/or by gating the clocks to the APB and AHB peripherals when they are unused.

The system operation mode depends on the CPU sub-system operating mode. The system only enters a low-power mode when the CPU allows it to do so.

The system low-power mode to enter depends on the allowed mode selected by the CPU in LPMS[2:0] bits in *PWR control register 1 (PWR_CR1)*.

Table 36. Low-power mode summary

Mode name	Entry	Wakeup source(1)	Wakeup system clock	Effect on clocks	Voltage regulators	
					MR	LPR
Sleep (Sleep-now or Sleep-on-exit)	WFI or return from ISR	Any interrupt	Same as before entering Sleep mode	CPU clock OFF No effect on other clocks or analog clock sources	ON	ON
	WFE	Wakeup event				
LPRun	Set LPR bit	Clear LPR bit	Same as LPRun clock	None	OFF	ON
LPSleep	Set LPR bit + WFI or return from ISR	Any interrupt	Same as before entering LPSleep mode	CPU clock OFF No effect on other clocks or analog clock sources	OFF	ON
	Set LPR bit + WFE	Wakeup event			OFF	ON

Table 36. Low-power mode summary (continued)

Mode name	Entry	Wakeup source(1)	Wakeup system clock	Effect on clocks	Voltage regulators	
					MR	LPR
Stop 0	LPMS = 0b000 + SLEEPDEEP bit + WFI or return from ISR or WFE	Any EXTI line (configured in the EXTI registers). Specific peripherals events	HSI16 when STOPWUCK = 1 in RCC_CFGR. MSI with the frequency before entering the Stop mode when STOPWUCK = 0.	All clocks OFF except HSI16, LSI and LSE	ON	ON
Stop 1	LPMS = 0b001 + SLEEPDEEP bit + WFI or return from ISR or WFE				OFF	
Stop 2 (with I2C3, LPUART1, LPTIM1, SRAM1, SRAM2)	LPMS = 0b010+ SLEEPDEEP bit + WFI or return from ISR or WFE					
Standby (with SRAM2)	LPMS = 0b011+ Set RRS bit + SLEEPDEEP bit + WFI or return from ISR or WFE	Wakeup PVD, RFIRQ, wakeup RFBUSY, WKUP pin edge, RTC and TAMP event, LSECSS, external reset in NRST pin, IWDG reset	MSI 4 MHz	All clocks OFF except LSI and LSE		
Standby	LPMS = 0b011 + Clear RRS bit + SLEEPDEEP bit + WFI or return from ISR or WFE				OFF	OFF
Shutdown	LPMS = 0b1xx + SLEEPDEEP bit + WFI or return from ISR or WFE	WKUP pin edge, RTC and TAMP event, external reset in NRST pin	MSI 4 MHz	All clocks OFF except LSE	OFF	OFF

1. Refer to *Table 37: Functionalities depending on system operating mode*.

Table 37. Functionalities depending on system operating mode(1)

Peripheral	Run	Sleep	LPRun	LPSleep	Stop 0		Stop 1		Stop 2		Standby		Shutdo		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
CPU	Y	-	Y	-	R	-	R	-	R	-	-	-	-	-	-
Radio-system (sub-GHz)	O	O	O	O	O	O	O	O	O	O	O	O	-	-	-

Table 37. Functionalities depending on system operating mode[1] (continued)

Peripheral	Run	Sleep	LPRun	LPSleep	Stop 0		Stop 1		Stop 2		Standby		Shutdo		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
Flash memory (up to 256 Kbytes)	Y	O[2]	O[3]	O[2] [3]	R	-	R	-	R	-	R	-	R	-	R
Flash memory interface	Y	Y	Y	Y	R	-	R	-	R	-	-	-	-	-	-
SRAM1	Y	O[2]	Y	O[2]	R	-	R	-	R	-	-	-	-	-	-
SRAM2	Y	O[2]	Y	O[2]	R	-	R	-	R	-	O[4]	-	-	-	-
Backup registers	Y	Y	Y	Y	R	-	R	-	R	-	R	-	R	-	R
Brownout reset (BOR)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-
Programmable voltage detector (PVD)	O	O	O	O	O	O	O	O	O	O	O[5]	O[5]	-	-	-
Peripheral voltage monitor (PVM3)	O	O	O	O	O	O	O	O	O	O	-	-	-	-	-
DMAx (x = 1, 2)	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
DMAMUX1	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
High-speed internal (HSI16)	O	O	O	O	O[6]	-	O[6]	-	O[6]	-	-	-	-	-	-
High-speed external (HSE32)	O	O	O[7]	O[7]	O[7]	-	O[7]	-	O[7]	-	O[7]	-	-	-	-
Low-speed internal (LSI)	O	O	O	O	O	-	O	-	O	-	O	-	-	-	-
Low-speed external (LSE)	O	O	O	O	O	-	O	-	O	-	O	-	O	-	O
Multi-speed internal (MSI)	O	O	O	O	O	-	O	-	O	-	-	-	-	-	-
Clock security system (CSS)	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
Clock security system on LSE	O	O	O	O	O	O	O	O	O	O	O	O	-	-	-
RTC/auto wakeup	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
Number of TAMP tamper pins	3	3	3	3	3	O	3	O	3	O	3	O	3	O	3
USARTx (x= 1, 2)	O	O	O	O	O[8]	O[8]	O[8]	O[8]	-	-	-	-	-	-	-
Low-power UART (LPUART1)	O	O	O	O	O[8]	O[8]	O[8]	O[8]	O[8]	O[8]	-	-	-	-	-
I2Cx (x = 1, 2)	O	O	O	O	O[9]	O[9]	O[9]	O[9]	-	-	-	-	-	-	-
I2C3	O	O	O	O	O[9]	O[9]	O[9]	O[9]	O[9]	O[9]	-	-	-	-	-
SPI1	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
SUBGHZSPI	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
SPI2S2	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
ADC	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
DAC	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-

Table 37. Functionalities depending on system operating mode[1] (continued)

Peripheral	Run	Sleep	LPRun	LPSleep	Stop 0		Stop 1		Stop 2		Standby		Shutdo		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
VREFBUF	O	O	O	O	O	-	O	-	R	-	-	-	-	-	-
COMPx (x = 1, 2)	O	O	O	O	O	O	O	O	O	O	-	-	-	-	-
Temperature sensor	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
Timers (TIMx) x = 1, 2, 16, 17)	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
LPTIM1	O	O	O	O	O	O	O	O	O	O	-	-	-	-	-
LPTIMx (x = 2, 3)	O	O	O	O	O	O	O	O	-	-	-	-	-	-	-
Independent watchdog (IWDG)	O	O	O	O	O	O	O	O	O	O	O	O	-	-	-
Window watchdog (WWDG)	O	O	O	O	R	-	R	-	R	-	-	-	-	-	-
SysTick timer	O	O	O	O	R	-	R	-	R	-	-	-	-	-	-
True random number generator (RNG)	O (10)	O (10)	R	R	R	-	R	-	-	-	-	-	-	-	-
AES hardware accelerator	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
PKA hardware accelerator	O	O	O	O	R	-	R	-	-	-	-	-	-	-	-
CRC calculation unit	O	O	O	O	R	-	R	-	R	-	-	-	-	-	-
HSEM	O	R	O	R	R	-	R	-	-	-	-	-	-	-	-
EXTI	O	O	O	O	R	O	R	O	R	O	-	-	-	-	-
GPIOs	O	O	O	O	O	O	O	O	O	O	R[11]	3 pins (12)	(13)	3 pins[12]	-

1. Legend: Y = Yes (enable). O = Optional (disable by default and can be enabled by software). R = data retained. - = Not available. Gray cells indicate wakeup capability.
2. The SRAM clock can be gated on or off.
3. Flash memory can be placed in power-down mode.
4. The SRAM2 content can optionally be retained when the PWR_CR3.RRS bit is set.
5. Only when the sub-GHz radio is active.
6. Some peripherals with wakeup from Stop capability can request HSI16 to be enabled. In this case, HSI16 is woken up by the peripheral, and only feeds the peripheral that requested it. HSI16 is automatically put off when the peripheral does not need it anymore.
7. HSE32 can be used by sub-GHz radio system.
8. UART reception is functional in Stop 0 and 1 modes. LPUART1 reception is functional is Stop 0, 1, and 2 modes. LPUART1 generates a wakeup interrupt on Start address match or received frame event.
9. I2Cx (x= 1, 2) address detection is functional in Stop 0 and 1 modes. I2C3 address detection is functional in Stop 0, 1, and 2 modes. I2C3 generates a wakeup interrupt in case of address match.
10. Voltage scaling range 1 only.

11. I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
12. The I/Os with wakeup from Standby/Shutdown capability are PA0, PC13 and PB3.
13. I/Os can be configured with internal pull-up, pull-down or floating in Shutdown mode, but the configuration is lost when exiting the Shutdown mode.

Relation between MCU and sub-GHz radio operating modes

The CPU and sub-GHz radio have their own operating modes, as defined in the table below.

Table 38. MCU and sub-GHz radio operating modes

CPU operating mode	Sub-GHz radio operating mode(1)	Description
Run, Sleep	Sleep, Calibration, Standby, Active (FS, TX, RX)(2)	LDO or SMPS regulator active, MCU running in main regulator (MR) mode.
LPRun, LPSleep	Deep-Sleep	LDO and SMPS regulator off, MCU running in low power regulator (LPR) mode.
	Sleep, Calibration, Standby, Active (FS, TX, RX)	LDO or SMPS regulator active, MCU running in low power regulator (LPR) mode.
Stop 0	Sleep, Calibration, Standby, Active (FS, TX, RX)(2)	LDO or SMPS regulator active, MCU running in main regulator (MR) mode.
Stop 1 and Stop 2	Deep-Sleep	LDO and SMPS regulator off, MCU using low power regulator (LPR) mode.
	Sleep, Calibration, Standby, Active (FS, TX, RX)	LDO or SMPS regulator active, MCU using low power regulator (LPR) mode.
Standby	Deep-Sleep	LDO and SMPS regulator off, MCU regulator off or on in low power (LPR) mode(3).
	Sleep, Calibration, Standby, Active (FS, TX, RX)	LDO or SMPS regulator active, MCU regulator off or on in low power (LPR) mode(3).
Shutdown	Deep-Sleep(4)	LDO and SMPS regulator off, MCU regulator off

1. For more details on sub-GHz radio operating modes, see *Section 4: Sub-GHz radio (SUBGHZ)*.
2. In the MCU Run, Sleep and Stop 0 modes, the sub-GHz radio is prevented from entering Deep-Sleep mode.
3. When retaining SRAM2 in Standby mode, the MCU uses the low-power regulator (LPR) mode.
4. When the CPU is in Shutdown mode, the sub-GHz radio cannot be activated and is forced in Deep-Sleep mode.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop 0, Stop 1, Stop 2, Standby or Shutdown mode while the debug features are used. This is because the CPU core is no longer clocked.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Section 36.3.6: Serial-wire and JTAG debug port*.

In Stop 0 and 1 modes, the EXTI CDBGPWRUPREQ wakeup event can be used to restart the CPU clock by the debugger. For more details, refer to *Section 14.3.1: EXTI wakeup interrupt list*.

5.4.1 Run mode

Slowing down system clocks

In Run mode, the speed of the system clocks (SYSCLK, HCLK, PCLK) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down the peripherals before entering the Sleep mode.

For more details, refer to *RCC clock configuration register (RCC_CFGR)*.

Peripheral clock gating

In Run mode, HCLK and PCLK for individual peripherals and memories can be stopped at any time to reduce the power consumption.

In Sleep mode, to further reduce the power consumption, the peripheral clocks can be disabled prior to executing the WFI or WFE instructions.

The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

Disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bits in the RCC_AHBxSMENR and RCC_APBxSMENR registers.

5.4.2 Low-power run mode (LPRun)

To further reduce the consumption when the system is in Run mode, the regulator can be configured in low-power mode. In this mode, the HCLK bus frequency must not exceed 2 MHz. HPRE and SHDHPRE must be used to divide the SYSCLK frequency (or MSI not exceeding 2 MHz must be used) before entering LPRun mode.

Warning: In LPRun mode, HSE32 cannot be used and must be disabled before entering LPRun mode.

In LPRun mode HSI16 can be used as kernel clock for peripherals and PLL must be disabled.

The device only enters LPRun mode once the low-power regulator is ready. The REGLPS bit can be used to check that the low-power regulator is ready. The REGLPF bit must be used to know if the device is in LPRun mode.

Refer to the product datasheet for more details on voltage regulator and peripherals operating conditions.

I/O states in LPRun mode

In LPRun mode, all I/O pins keep the same state as in Run mode.

Enter LPRun mode

To enter the LPRun mode, proceed as follows (refer to *Table 39*):

1. Jump into the SRAM and power down the flash memory by setting the FPDR bit in the *Section 5.5.1: PWR control register 1 (PWR_CR1)* (optional).
2. Disable HSE32 clock.
3. Decrease the HCLK clock frequencies below 2 MHz.
4. Force the regulator in low-power mode by setting the LPR bit in the *PWR control register 1 (PWR_CR1)*.

Exit LPRun mode

To exit the LPRun mode, proceed as follows (refer to *Table 39*):

1. Force the regulator in main mode by clearing the LPR bit in the *PWR control register 1 (PWR_CR1)*.
2. Wait until REGLPF bit is cleared in the *Power status register 2 (PWR_SR2)*.
3. Increase the HCLK clock frequency (enable HSE32 clock when needed).

Table 39. LPRun

LPRun mode	Description
Mode entry	Decrease the system clock frequency below 2 MHz. LPR = 1.
Mode exit	LPR = 0. Wait until REGLPF = 0. Increase the system clock frequency.
Wakeup latency	Regulator wakeup time from low-power mode

5.4.3 Enter low-power mode

The MCU enters low-power mode following one of these events:

- when MCU executes the WFI (wait for interrupt)
- when MCU executes WFE (wait for event) instructions
- on return from ISR when the SLEEPONEXIT bit in the CPU system control register is set

Low-power mode is only be entered if no interrupt or event is pending.

5.4.4 Exit low-power mode

From Sleep and Stop modes, the CPU exits low-power mode depending on the way the mode was entered, as detailed below:

- If the WFI instruction or return from ISR was used to enter the low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the low-power mode, the CPU exits the low-power mode as soon as an event occurs. The wakeup event can be generated either by an NVIC IRQ interrupt or by an event.
  - **Wakeup generated by an NVIC IRQ with SEVONPEND = 0 in the CPU system control register, enabling an interrupt in the peripheral control register and in the NVIC)**
    When the CPU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending

register) must be cleared. Only NVIC interrupts with sufficient priority wake up and interrupt the CPU.

– **Wakeup generated by an NVIC IRQ with SEVONPEND = 1 in the CPU system control register, enabling an interrupt in the peripheral control register and optionally in the NVIC**
When the CPU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. All NVIC interrupts wake up the CPU, even the disabled ones. Only enabled NVIC interrupts with sufficient priority wake up and interrupt the CPU.

– **Wakeup generated by an event, configuring an EXTI line in event mode**
When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set. It may be necessary to clear the interrupt flag in the peripheral.

From Standby and Shutdown modes, the CPU exits low-power mode through an external reset (NRST pin), an IWDG reset, a sub-GHz radio IRQ, a PVD event, an edge on the sub-GHz radio busy signal, a edge on one of the enabled WKUPx pins, an RTC and TAMP event, or a radio event (for Standby only).

After waking up from Standby or Shutdown mode, the program execution restarts in the same way as after a reset (boot pin sampling, option bytes loading, reset vector is fetched).

The system mode when it wakes up from low-power mode can be determined from the C1STOPF, C1STOP2F and C1SBF bits in the *PWR extended status and status clear register (PWR_EXTSCR)*.

Table 40. CPU wakeup versus system operating mode

System mode	CPU			CPU wakeup
	C1SBF	C1STOPF	C1STOP2F	
Run	0	0	0	Kept running
Stop	0	1	0	Wakeup from Stop 0 or 1
	0	0	1	Wakeup from Stop 2
Standby	1	0	0	Wakeup from Standby
N/A	Others			Not valid, does not occur

5.4.5 Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

Enter Sleep mode

The Sleep mode is entered from Run mode according to *Enter low-power mode*, when the SLEEPDEEP bit in the CPU system control register is cleared (see *Table 41)*.

Exit Sleep mode

The MCU exits the Sleep mode (see *Table 41)* as indicated in *Exit low-power mode*.

Table 41. Sleep mode

Sleep mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) is pending Refer to the Cortex system control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex system control register.
Mode exit	If WFI or return from ISR was used for entry Interrupt: refer to *Table 78: Vector table* If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to *Table 81: Wakeup interrupts* If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to *Table 78: Vector table* or wakeup event: refer to *Table 81: Wakeup interrupts*
Wakeup latency	None

5.4.6 Low-power sleep mode (LPSleep)

Refer to the product datasheet for more details on voltage regulator and peripherals operating conditions.

I/O states in LPSleep mode

In LPSleep mode, all I/O pins keep the same state as in Run mode.

Enter LPSleep mode

The LPSleep mode is entered from LPRun mode as described in *Section 5.4.3: Enter low-power mode*, when the SLEEPDEEP bit in the Cortex system control register is clear.

Refer to *Table 42* for details on how to enter the LPSleep mode.

Exit LPSleep mode

The LPSleep mode is exited as described in *Section 5.4.4: Exit low-power mode*. When exiting the LPSleep mode by issuing an interrupt or an event, the MCU is in LPRun mode.

The table below details how to exit the LPSleep mode.

Table 42. LPSleep

LPSleep mode	Description
Mode entry	LPSleep mode is entered from the LPRun mode. WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) is pending Refer to the Cortex system control register.
	LPSleep mode is entered from the LPRun mode. On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex system control register.
Mode exit	If WFI or return from ISR was used for entry Interrupt: refer to *Table 78: Vector table* If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to *Table 81: Wakeup interrupts* If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to *Table 78: Vector table* Wakeup event: refer to *Table 81: Wakeup interrupts* After exiting the LPSleep mode, the MCU is in LPRun mode.
Wakeup latency	None

5.4.7 Stop 0 mode

The Stop 0 mode is based on the CPU Deep-Sleep mode combined with the peripheral clock gating. The voltage regulator is configured in main regulator mode. In Stop 0 mode, all clocks in the V_{CORE} domain are stopped.PLL, MSI, HSI16 and HSE32 oscillators are disabled. Some peripherals with the wakeup capability (I2Cx (x = 1, 3), USARTx (x = 1, 2) and LPUART1) can switch on HSI16 to receive a frame, and switch off HSI16 after receiving the frame if it is not a wakeup frame. In this case, the HSI16 clock is propagated only to the peripheral requesting it.

SRAM1, SRAM2 and register contents are preserved.

The BOR is always available in Stop 0 mode. The consumption is increased when thresholds higher than V_{BOR0} are used.

The BOR and PDR can be activated to sample periodically the supply voltage. This option enabled by setting the ULPEN bit of the PWR_CR3 register allows decreasing the current consumption in this mode, but any drop of the voltage below the operating conditions between two active periods of the supply detector results in a non-generation of PDR reset.

I/O states in Stop 0 mode

In the Stop 0 mode, all I/O pins keep the same state as in the Run mode.

Enter Stop 0 mode

The Stop 0 mode is entered according *Section 5.4.3*, when the SLEEPDEEP bit in the Cortex system control register is set (see *Table 43*).

If flash memory programming is ongoing, the Stop 0 mode entry is delayed until the operation is completed.

If an access to the APB domain is ongoing, the Stop 0 mode entry is delayed until the APB access is finished.

In Stop 0 mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its key register or by hardware option. Once started, it cannot be stopped except by a reset. See *Section 28.3: IWDG functional description*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the *RCC Backup domain control register (RCC_BDCR)*.
- Internal RC oscillator (LSI): this is configured by the LSIxON bit in the *RCC control/status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the *RCC Backup domain control register (RCC_BDCR)*.
- Sub-GHz radio activity, as programmed, see *Section 4: Sub-GHz radio (SUBGHZ)*.
- PVD detection configured in *PWR control register 3 (PWR_CR3)*.

Several peripherals can be used in Stop 0 mode and can add consumption if they are enabled and clocked by LSI or LSE: LPTIMx (x = 1, 2, 3), I2Cx (x = 1, 2, 3), USARTx (x = 1, 2), LPUART1.

In Stop 0 mode, when HSIKERON is enabled, the wakeup capabilities of some peripherals are also available when clocked by HSI16: I2Cx (x = 1, 2, 3), USARTx (x = 1, 2) or LPUART1.

The comparators can be used in Stop 0 mode, PVM3 and PVD as well. If they are not needed, they must be disabled by software to save their power consumption.

ADC, temperature sensor and VREFBUF buffer can consume power during the Stop 0 mode, unless they are disabled before entering this mode.

Exit Stop 0 mode

The Stop 0 mode is exited according to what is indicated in *Section 5.4.4* (see *Table 43* for details).

When exiting Stop 0 mode by issuing an interrupt or a wakeup event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in *RCC clock configuration register (RCC_CFGR)*. The MSI oscillator is selected as system clock if the bit STOPWUCK is cleared. The wakeup time is shorter when HSI16 is selected as wakeup system clock. The MSI selection enables a wakeup at higher frequency, up to 48 MHz.

When the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop 0 mode with HSI16. By keeping the internal regulator on during Stop 0 mode, the consumption is higher but the startup time is reduced.

When exiting the Stop 0 mode, the MCU is either in Run mode (range 1 or range 2 depending on VOS bit in *PWR control register 1 (PWR_CR1)*) or in LPRun mode if the bit LPR is set in the same register.

Table 43. Stop 0 mode

Stop 0	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex system control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = 0b000 in PWR_CR1
	On return from ISR while: – SLEEPDEEP bit is set in Cortex system control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = 0b000 in PWR_CR1
	Note: To enter Stop 0 mode, all EXTI line pending bits (in EXTI pending register (EXTI_PR1)), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and program execution continues.
Mode exit	If WFI or return from ISR was used for entry Any EXTI line configured in Interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to *Table 78: Vector table*. If WFE was used for entry and SEVONPEND = 0: Any EXTI line configured in event mode. Refer to *Table 81: Wakeup interrupts*. If WFE was used for entry and SEVONPEND = 1: Any EXTI line configured in Interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to*Table 78: Vector table*. Wakeup event: refer to *Table 81: Wakeup interrupts*.
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and flash memory wakeup time from Stop 0 mode.

5.4.8 Stop 1 mode

The Stop 1 mode is the same as Stop 0 mode except that the main regulator is off, and only the low-power regulator is on. Stop 1 mode can be entered from Run mode and from LPRun mode. The device only enters Stop 1 mode once the low-power regulator is ready. The

REGLPS bit can be used to check that the low-power regulator is ready (see the table below).

Table 44. Stop 1 mode

Stop 1	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex system control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = 0b001 in PWR_CR1
	On return from ISR while: – SLEEPDEEP bit is set in Cortex system control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = 0b001 in PWR_CR1
	Note: To enter Stop 1 mode, all EXTI line pending bits (in EXTI pending register (EXTI_PR1)), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop 1 mode entry procedure is ignored and program execution continues.
Mode exit	If WFI or return from ISR was used for entry Any EXTI line configured in Interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to *Table 78: Vector table*. If WFE was used for entry and SEVONPEND = 0: Any EXTI line configured in event mode. Refer to *Section 14.4.1: EXTI configurable event input wakeup*. If WFE was used for entry and SEVONPEND = 1: Any EXTI line configured in Interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to *Table 78: Vector table*. Wakeup event: refer to *Table 81: Wakeup interrupts*
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and regulator wakeup time from Low-power mode + flash memory wakeup time from Stop 1 mode.

5.4.9 Stop 2 mode

The Stop 2 mode is based on the CPU Deep-Sleep mode combined with peripheral clock gating and partial power down of the V_{CORE} domain. In Stop 2 mode, all clocks in the V_{CORE} domain are stopped. PLL, MSI, HSI16 and HSE32 oscillators are disabled. Some of the logic is powered down except for the CPU and some peripherals with wakeup capability (I2C3, LPTIM1 and LPUART1), that can switch on the HSI16 to receive a frame, and switch off HSI16 after receiving the frame if it is not a wakeup frame. In this case, the HSI16 clock is propagated only to the peripheral requesting it. The Stop 2 mode uses the low-power regulator, hence the device only enters Stop 2 mode once the low-power regulator is ready. The REGLPS bit can be used to check that the low-power regulator is ready.

SRAM1, SRAM2, PWR, flash memory interface, RCC, EXTI, IWDG, WWDG, GPIO, CRC, SYSCFG, RTC and TAMP contents and registers in the Backup domain are also preserved. The content of all other peripherals is reset and must be reprogrammed.

The BOR is always available in Stop 2 mode. The consumption is increased when thresholds higher than V_{BOR0} are used.

The BOR and PDR can be activated to sample periodically the supply voltage. This option enabled by setting the ULPEN bit of the PWR_CR3 register allows the current consumption to be decreased in this mode, but any drop of the voltage below the operating conditions between two active periods of the supply detector, results in a non-generation of PDR reset.

Note: The comparators, LPUART1 and LPTIM1 outputs are forced to low speed (OSPEEDy = 0b00) during the Standby mode.

I/O states in Stop 2 mode

In Stop 2 mode, all I/O pins keep the same state as in the Run mode.

Enter Stop 2 mode

The Stop 2 mode is entered as described in *Section 5.4.3*, when the SLEEPDEEP bit in the Cortex system control register is set (see *Table 45*).

Stop 2 mode can only be entered from Run mode. It is not possible to enter Stop 2 mode from the LPRun mode.

If flash memory programming is ongoing, the Stop 2 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop 2 mode entry is delayed until the APB access is finished.

Several peripherals can be used in Stop 2 mode and can add consumption if they are enabled and clocked by LSI or LSE: LPTIM1, I2C3 and LPUART1.

In Stop 2 mode, when HSIKERON is enabled, the wakeup capabilities of some peripherals are also available when clocked by HSI16: I2C3 or LPUART1.

The comparators can be used in Stop 2 mode, PVM3 and PVD as well. If they are not needed, they must be disabled by software to save their power consumption.

ADC, temperature sensor and VREFBUF buffer can consume power during Stop 2 mode, unless they are disabled before entering this mode.

All the peripherals that cannot be used in Stop 2 mode are powered down.

Exit Stop 2 mode

The Stop 2 mode is exited according *Section 5.4.4* (see *Table 45*).
When exiting Stop 2 mode by issuing an interrupt or a wakeup event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in *RCC clock configuration register (RCC_CFGR)*. The MSI oscillator is selected as system clock if the bit STOPWUCK is cleared. The wakeup time is shorter when HSI16 is selected as wakeup system clock. The MSI selection allows a wakeup at higher frequency, up to 48 MHz,

The STOP2F status flag in the *PWR control register 3 (PWR_CR3)* indicates that the MCU was in Stop 2 mode. All non retained registers are reset after wakeup from Stop 2. When exiting the Stop 2 mode, the MCU is in Run mode (range 1 or range 2 depending on VOS bit in PWR_CR1).

Table 45. Stop 2 mode

Stop 2	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex system control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = 0b010 in PWR_CR1
	On return from ISR while: – SLEEPDEEP bit is set in Cortex system control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = 0b010 in PWR_CR1
	Note: To enter Stop 2 mode, all EXTI line pending bits in EXTI pending register (EXTI_PR1), and EXTI pending register (EXTI_PR2)), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Standby mode entry procedure is ignored and program execution continues.
Mode exit	Any EXTI line configured in Interrupt mode or event mode (regardless if the corresponding EXTI interrupt vector is enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to *Table 78: Vector table*.
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and regulator wakeup time from Low-power mode + flash wakeup time from Stop 2 mode.

5.4.10 Standby mode

The Standby mode allows the lowest power consumption to be achieved with BOR. It is based on the CPU Deep-Sleep mode, with the voltage regulators disabled (except when SRAM2 content is preserved). PLL, HSI16, MSI and HSE32 oscillators are also switched off.

SRAM1 and register contents are lost except for registers in the Backup domain and Standby circuitry (see *Figure 14*). SRAM2 content can be preserved if the bit RRS is set in the *PWR control register 3 (PWR_CR3)*. In this case the low-power regulator is enabled and provides the supply to SRAM2 only. When the SRAM2 retention is enabled the device only enters Standby mode once the low-power regulator is ready. The REGLPS bit can be used to check that the low-power regulator is ready. Immediately exiting from Standby mode with SRAM2 retention may be delayed until the low-power regulator is ready.

BOR is always available in Standby modes. The consumption is increased when thresholds higher than V_{BOR0} are used.

BOR and PDR can be activated to sample periodically the supply voltage. This option enabled by setting the ULPEN bit of the PWR_CR3 register allows the current consumption to be decreased in this mode, but any drop of the voltage below the operating conditions between two active periods of the supply detector results in a non-generation of PDR reset.

I/O states in Standby mode

In Standby mode, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x = A, B, C, H)), or with a pull-down (refer to PWR_PDCRx registers (x = A, B, C, H)), or can be kept in analog state.

The RTC output on PC13 only is functional in Standby mode. PC14 and PC15 used for LSE are also functional. Three wakeup pins (WKUPx, x = 1, 2, 3) and the three TAMP tampers are available.

The sub-GHz radio is functional and PVD can be enabled.

Enter Standby mode

The Standby mode is entered according *Section 5.4.3*, when the SLEEPDEEP bit in the Cortex system control register is set (see *Table 46* for details).

In Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its key register or by hardware option. Once started it cannot be stopped except by a reset. See *Section 28.3: IWDG functional description*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR)
- Internal RC oscillator (LSI): this is configured by the LSIxON bit in the control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)
- Sub-GHz radio activity, as programmed, see *Section 4: Sub-GHz radio (SUBGHZ)*.
- PVD detection configured in *PWR control register 3 (PWR_CR3)*.

Exit Standby mode

The Standby mode is exited according to *Section 5.4.4*. The SBF status flags (CnSBF) in the *PWR extended status and status clear register (PWR_EXTSCR)* indicate that the MCU was in Standby mode. All registers are reset after wakeup from Standby except for *PWR extended status and status clear register (PWR_EXTSCR)*.

Refer to the table below for more details on how to exit Standby mode.

Table 46. Standby mode

Standby	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex system control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = 0b011 in PWR_CR1 – WUFx bits are cleared in power status register 1 (PWR_SR1)
	On return from ISR while: – SLEEPDEEP bit is set in Cortex system control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = 0b011 in PWR_CR1 – Radio IRQ is cleared in the sub-GHz radio. – WPVDF, WRFBUSY and WUFx bits are cleared in power status register 1 (PWR_SR1) – RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, synchronous binary counter or timestamp flags) is cleared – TAMP flags ITAMPxF and TAMPxF are cleared.
Mode exit	PVD event, RFIRQ interrupt, RFBUSY wakeup event, WKUPx pin edge, RTC and TAMP event, external reset in NRST pin, IWDG reset, BOR reset
Wakeup latency	Reset phase

5.4.11 Shutdown mode

The lowest power consumption can be reached in Shutdown mode. It is based on the Deep-Sleep mode, with the voltage regulator disabled. The V_{CORE} domain is consequently powered off. PLL, HSI16, MSI, LSI and HSE32 oscillators are also switched off.

SRAM1, SRAM2 and registers contents are lost except for registers in the Backup domain. BOR is not available in Shutdown mode. No power voltage monitoring is possible in this mode, therefore the switch to the Backup domain is not supported.

I/O states in Shutdown mode

In Shutdown mode, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x = A, B, C, H), or with a pull-down (refer to PWR_PDCRx registers (x = A, B, C, H)), or can be kept in analog state. However this configuration is lost when exiting the Shutdown mode due to the power-on reset.

The RTC output on PC13 only is functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. Three wakeup pins (WKUPx, x = 1, 2, 3) and the three TAMP tampers are available.

Enter Shutdown mode

The Shutdown mode is entered according *Section 5.4.3*, when the SLEEPDEEP bit in the Cortex system control register is set (see *Table 47* for details).

In Shutdown mode, the following features can be selected by programming individual control bits:

- Real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR).

Caution: In case of V_{DD} power-down, the RTC content is lost.

- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)

Exit Shutdown mode

The Shutdown mode is exited according *Section 5.4.4*. A power-on reset occurs when exiting from Shutdown mode. All registers (except for the ones in the Backup domain) are reset after wakeup from Shutdown.

Refer to the table below for more details on how to exit Shutdown mode.

Table 47. Shutdown mode

Shutdown mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex system control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = 0b1xx in PWR_CR1 – WUFx bits are cleared in power status register 1 (PWR_SR1)
	On return from ISR while: – SLEEPDEEP bit is set in Cortex system control register – SLEEPONEXT = 1 – No interrupt is pending – LPMS = 0b1xx in PWR_CR1 – Radio IRQ is cleared in the sub-GHz radio. – WPVDF, WRFBUSY and WUFx bits are cleared in power status register 1 (PWR_SR1) – The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, synchronous binary counter or timestamp flags) is cleared – TAMP flags ITAMPxF and TAMPxF are cleared.
Mode exit	WKUPx pin edge, RTC and TAMP event, external reset in NRST pin
Wakeup latency	Reset phase

5.4.12 Auto-wakeup from low-power mode

The RTC can be used to wake up the MCU from low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop (0, 1 or 2) or Standby mode at regular intervals. For this purpose, the

following alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the *RCC Backup domain control register (RCC_BDCR)*:

- Low-power 32.768 kHz external crystal oscillator (LSE OSC)

  This clock source provides a precise time base with very low-power consumption.
- Low-power internal RC oscillator (LSI)

  This clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC oscillator is designed to add minimum power consumption.

To wakeup from Stop mode with an RTC alarm event, it is necessary to:

- Configure the EXTI line 18 to be sensitive to rising edge.
- Configure the RTC to generate the RTC alarm.

To wakeup from Standby mode, there is no need to configure the EXTI line 18.

To wakeup from Stop mode with an RTC wakeup event, it is necessary to:

- Configure the EXTI line 20 to be sensitive to rising edge.
- Configure the RTC to generate the RTC alarm.

To wakeup from Standby mode, there is no need to configure the EXTI line 20.

5.5 PWR registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

5.5.1 PWR control register 1 (PWR_CR1)

This register is reset after wakeup from Standby mode, except for bit field LPMS[2:0].

Access: additional APB cycles are used to access this register versus those used for a standard APB access (three for a write and two for a read).

Address offset: 0x000

Reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPR	Res.	Res.	Res.	VOS[1:0]		DBP	Res.	Res.	FPDS	FPDR	SUBGHZSPINSSSEL	LPMS[2:0]		
	rw				rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **LPR**: LPRun

When this bit is set, the supply mode is switched from main regulator mode (MR) to low-power regulator mode (LPR).

Note: Stop 2 mode cannot be entered when LPR bit is set. Stop 1 is entered instead.

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:9 **VOS[1:0]**: Voltage scaling range selection

Previous voltage range change must be completed before chaining voltage range again.
0b00: Cannot be written (forbidden by hardware)
0b01: Range 1
0b10: Range 2
0b11: Cannot be written (forbidden by hardware)

Bit 8 **DBP**: Disable Backup domain write protection

In reset state, the RTC and backup registers are protected against parasitic write access. This bit must be set to enable write access to these registers.
0: Access to RTC and backup registers disabled
1: Access to RTC and backup registers enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **FPDS**: Flash memory power-down mode during LPSleep

This bit selects whether the flash memory is in Power-down mode or Idle mode when the CPU is in Sleep mode. flash memory is only set in power-down mode when the system is in LPSleep mode.
0: Flash memory in Idle mode when system is in LPSleep mode
1: Flash memory in Power-down mode when system is in LPSleep mode

Bit 4 **FPDR**: Flash memory power-down mode during LPRun

This bit can only be written to 1 after unlocking this register bit, by first writing (code 0xC1B0) into this register (when writing the code, the register bits are not updated). Selects whether the flash memory is in power-down mode or Idle mode when in LPRun mode. (flash memory can only be in power-down mode when code is executed from SRAM). Flash memory is only set in power-down mode when the system is in LPRun mode.

0: Flash memory in Idle mode when system is in LPRun mode

1: Flash memory in Power-down mode when system is in LPRun mode

Bit 3 **SUBGHZSPINSSSEL**: sub-GHz SPI NSS source select

This bit is set and cleared by software.

0: sub-GHz SPI NSS signal driven from PWR_SUBGHZSPICR.NSS (RFBUSYMS functionality enabled)

1: sub-GHz SPI NSS signal driven from LPTIM3_OUT (RFBUSYMS functionality disabled)

Bits 2:0 **LPMS[2:0]**: Low-power mode selection

These bits are not reset when exiting Standby mode.

These bits select the low-power mode allowed when CPU enters the Deep -Sleep mode.

000: Stop 0 mode

001: Stop 1 mode

010: Stop 2 mode

011: Standby mode

1xx: Shutdown mode

Note: If LPR bit is set, Stop 2 mode cannot be selected and Stop 1 mode must be entered instead of Stop 2.

In Standby mode, SRAM2 is preserved, depending on RRS bit configuration in PWR_CR3.

5.5.2 PWR control register 2 (PWR_CR2)

This register is reset when exiting the Standby mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVME3	Res.	Res.	PLS[2:0]			PVDE
									rw			rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PVME3**: Peripheral voltage monitoring 3 enable: V_{DDA} versus 1.62 V

0: PVM3 (V_{DDA} monitoring versus 1.62 V threshold) disable.

1: PVM3 (V_{DDA} monitoring versus 1.62 V threshold) enable.

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:1 **PLS[2:0]**: Programmable voltage detector level selection.
These bits select the voltage threshold detected by the programmable voltage detector:
000: V_{PVD0} around 2.0 V
001: V_{PVD1} around 2.2 V
010: V_{PVD2} around 2.4 V
011: V_{PVD3} around 2.5 V
100: V_{PVD4} around 2.6 V
101: V_{PVD5} around 2.8 V
110: V_{PVD6} around 2.9 V
111: External input analog voltage PVD_IN (compared internally to VREFINT)

Note: These bits are write-protected when the bit PVDL (PVD Lock) is set in the SYSCFG_CBR register.
These bits are reset only by a system reset.

Bit 0 **PVDE**: Programmable voltage detector enable
0: Programmable voltage detector disabled
1: Programmable voltage detector enabled

Note: This bit is write-protected when the bit PVDL (PVD Lock) is set in the SYSCFG_CBR register.
This bit is reset only by a system reset.

5.5.3 PWR control register 3 (PWR_CR3)

This register is not reset when exiting Standby modes.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x008

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIWUL	Res.	EWRFIRQ	Res.	EWRFBUSY	APC	RRS	EWPVD	ULPEN	Res.	Res.	Res.	Res.	EWUP3	EWUP2	EWUP1
rw		rw		rw	rw	rw	rw	rw					rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EIWUL**: internal wakeup line for CPU enable
0: Internal wakeup line interrupt to CPU disabled
1: Internal wakeup line interrupt to CPU enabled

Bit 14 Reserved, must be kept at reset value.

Bit 13 **EWRFIRQ**: radio IRQ[2:0] wakeup for CPU enable
When this bit is set, the radio IRQ[2:0] is enabled and triggers a wakeup from Standby event to CPU.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **EWRFBUSY**: radio busy wakeup from Standby for CPU enable

When this bit is set, the radio busy is enabled and triggers a wakeup from Standby event to CPUwhen a rising or a falling edge occurs. The active edge is configured via the WRFBUSYP bit in the *PWR control register 4 (PWR_CR4)*.

Bit 10 **APC**: Apply pull-up and pull-down configuration from CPU

When this bit is set, the I/O pull-up and pull-down configurations defined in the PWR_PUCRx and PWR_PDCRx registers are applied. When this bit is cleared, the PWR_PUCRx and PWR_PDCRx registers are not applied to the I/Os.

Bit 9 **RRS**: SRAM2 retention in Standby mode

0: SRAM2 powered off in Standby mode (SRAM2 content lost)

1: SRAM2 powered by the low-power regulator in Standby mode (SRAM2 content kept)

Bit 8 **EWPVD**: PVD and wakeup for CPU enable (when sub-GHz radio in active state)

This bit is set and reset by software.

When this bit is set, the PVD is enabled while the sub-GHz radio is active and triggers an interrupt and wakeup from Stop, Standby event to CPU, when the voltage level drops below the PVD threshold level.

0: PVD not enabled by the sub-GHz radio active state

1: PVD enabled while the sub-GHz radio is active

Bit 7 **ULPEN:** Ultra-low-power enable

Enable/disable periodical sampling of supply voltage in Stop and Standby modes for detecting condition of PDR and BOR reset.

0: Disable (the supply voltage is monitored continuously)

1: Enable, when set, the supply voltage is sampled for PDR/BOR reset condition only periodically and not continuously, in order to save power. The sampling period is typically 12 ms, but is strongly linked to the temperature (the period decreases when the temperature increases).

Caution: When enabled and if the supply voltage drops below the minimum operating condition between two supply voltage samples, the reset condition is missed and no reset is generated.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **EWUP3**: wakeup pin WKUP3 for CPUenable

When this bit is set, the external wakeup pin WKUP3 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU. The active edge is configured via the WP3 bit in the *PWR control register 4 (PWR_CR4)*.

Bit 1 **EWUP2**: wakeup pin WKUP2 for CPUenable

When this bit is set, the external wakeup pin WKUP2 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU. The active edge is configured via the WP2 bit in the *PWR control register 4 (PWR_CR4)*.

Bit 0 **EWUP1**: wakeup pin WKUP1 for CPU enable

When this bit is set, the external wakeup pin WKUP1 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU. The active edge is configured via the WP1 bit in the *PWR control register 4 (PWR_CR4)*.

5.5.4 PWR control register 4 (PWR_CR4)

This register is not reset when exiting Standby modes.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WRFBUSYP	Res.	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	WP3	WP2	WP1
				rw		rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **WRFBUSYP**: Wakeup radio busy polarity
This bit defines the polarity used for an event detection on radio busy signal.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 10 Reserved, must be kept at reset value.

Bit 9 **VBRS**: V_{BAT} battery charging resistor selection
0: V_{BAT} charging through a 5 kΩ resistor
1: V_{BAT} charging through a 1.5 kΩ resistor

Bit 8 **VBE**: V_{BAT} battery charging enable
0: V_{BAT} battery charging disabled
1: V_{BAT} battery charging enabled

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **WP3**: Wakeup pin WKUP3 polarity
This bit defines the polarity used for an event detection on external wake-up pin, WKUP3
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 1 **WP2**: Wakeup pin WKUP2 polarity
This bit defines the polarity used for an event detection on external wake-up pin, WKUP2
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 0 **WP1**: Wakeup pin WKUP1 polarity
This bit defines the polarity used for an event detection on external wake-up pin, WKUP1
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

5.5.5 PWR status register 1 (PWR_SR1)

This register is not reset when exiting Standby modes.

Access: two additional APB cycles are needed to read this register versus a standard APB read.

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUFI	Res.	Res.	Res.	WRFBUSYF	Res.	Res.	WPVDF	Res.	Res.	Res.	Res.	Res.	WUF3	WUF2	WUF1
r				r			r						r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **WUFI**: Internal wakeup interrupt flag

This bit is set when a wakeup is detected on the internal wakeup line. It is cleared when all internal wakeup sources are cleared.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **WRFBUSYF**: Radio busy wakeup flag

This bit is set when a wakeup event is detected on radio busy. It is cleared by writing ‘1’ in the CWRFBUSYF bit of the PWR_SCR register.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **WPVDF**: Wakeup PVD flag

This bit is set when a wakeup event is detected on PVD. It is cleared by writing ‘1’ in the CWPVDF bit of the PWR_SCR register.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **WUF3**: Wakeup flag 3

This bit is set when a wakeup event is detected on wakeup pin, WKUP3. It is cleared by writing 1 in the CWUF3 bit of the *PWR status clear register (PWR_SCR)*.

Bit 1 **WUF2**: Wakeup flag 2

This bit is set when a wakeup event is detected on wakeup pin, WKUP2. It is cleared by writing 1 in the CWUF2 bit of the *PWR status clear register (PWR_SCR)*.

Bit 0 **WUF1**: Wakeup flag 1

This bit is set when a wakeup event is detected on wakeup pin, WKUP1. It is cleared by writing 1 in the CWUF1 bit of the *PWR status clear register (PWR_SCR)*.

5.5.6 Power status register 2 (PWR_SR2)

This register is partially reset when exiting Standby/Shutdown modes.

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PVMO3	Res.	Res.	PVDO	VOSF	REGLPF	REGLPS	FLASHRDY	REGMRS	RFEOLF	LDORDY	SMPSRDY	RFBUSYMS	RFBUSYS	Res.
	r			r	r	r	r	r	r	r	r	r	r	r	

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **PVMO3**: Peripheral voltage monitoring output: V_{DDA} versus 1.62 V

0: V_{DDA} voltage above PVM3 threshold (around 1.62 V)

1: V_{DDA} voltage below PVM3 threshold (around 1.62 V)

Note: PVMO3 is cleared when PVM3 is disabled (PVME3 = 0). After enabling PVM3, the PVM3 output is valid after the PVM3 wakeup time.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **PVDO**: Programmable voltage detector output

0: V_{DD} or voltage level on PVD_IN above the selected PVD threshold

1: V_{DD} or voltage level on PVD_IN below the selected PVD threshold

Bit 10 **VOSF**: Voltage scaling flag

A delay is required for the internal regulator to be ready after the voltage scaling has been changed. VOSF indicates that the regulator reached the voltage level defined with VOS bits of the *PWR control register 1 (PWR_CR1)*.

0: regulator ready in the selected voltage range

1: regulator output voltage changed to the required voltage level

Bit 9 **REGLPF**: low-power regulator flag

This bit is set by hardware when the MCU is in LPRun mode. When the MCU exits from the LPRun mode, this bit remains at 1 until the main regulator is ready. A polling on this bit must be done before increasing the product frequency.

This bit is cleared by hardware when the main regulator is ready.

0: main regulator (MR) ready and used

1: low-power regulator (LPR) used

Bit 8 **REGLPS**: low-power regulator started (ready)

This bit provides the information whether the low-power regulator is ready after a power-on reset or a Standby/Shutdown. If the Standby mode is entered while REGLPS bit is still cleared (“backup” SRAM2 disabled), the wakeup time from Standby mode may be increased.

0: LPR not ready

1: LPR ready

Bit 7 **FLASHRDY**: Flash memory ready

This bit is set by hardware when the flash memory can be accessed by software after a software controlled flash power down (in LPRun mode). This bit is cleared by hardware when the flash memory is powered down.

0: Flash memory not ready to be accessed

1: Flash memory ready to be accessed

Bit 6 **REGMRS**: Main regulator status

This bit is set by hardware when the main regulator is supplied by the LDO or SMPS when enabled. When this bit is cleared the main regulator is directly supplied from V_{DD}.

0: main regulator supplied directly from V_{DD}

1: main regulator supplied through LDO or SMPS

Bit 5 **RFEOLF**: Radio end-of-life flag

When enabled by RFEOLEBN, this bit indicates that the supply voltage reached the radio end-of-life operating low level.

0: Supply voltage above radio end-of-life operating low level

1: Supply voltage below radio end-of-life operating low level

Bit 4 **LDORDY**: LDO ready flag

This bit indicates that the LDO is ready.

0: LDO not ready or off

1: LDO ready

Bit 3 **SMPSRDY**: SMPS ready flag

This bit indicates that the SMPS step-down converter is ready.

0: SMPS step-down converter not ready or off

1: SMPS step-down converter ready

Bit 2 **RFBUSYMS**: Radio busy masked signal status

This bit indicates the actual status of the radio busy masked signal.

0: radio busy masked signal low (not busy)

1: radio busy masked signal high (busy)

Bit 1 **RFBUSYS**: Radio busy signal status

This bit indicates the actual status of the radio busy signal.

0: radio busy signal low (not busy)

1: radio busy signal high (busy)

Bit 0 Reserved, must be kept at reset value.

5.5.7 PWR status clear register (PWR_SCR)

Access: three additional APB cycles are needed to write this register versus a standard APB write.

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CWRFBUSYF	Res.	Res.	CWPVDF	Res.	Res.	Res.	Res.	Res.	CWUF3	CWUF2	CWUF1
				w			w						w	w	w

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **CWRFBUSYF**: Clear wakeup radio busy flag
Setting this bit clears the WRFBUSYF flag in the PWR_SR1. This bit is always read 0.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **CWPVDF**: Clear wakeup PVD interrupt flag
Setting this bit clears the WPVDF flag in the PWR_SR1. This bit is always read as 0.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **CWUF3**: Clear wakeup flag 3
Setting this bit clears the WUF3 flag in the PWR_SR1 register. This bit is always read as 0.

Bit 1 **CWUF2**: Clear wakeup flag 2
Setting this bit clears the WUF2 flag in the PWR_SR1 register. This bit is always read as 0.

Bit 0 **CWUF1**: Clear wakeup flag 1
Setting this bit clears the WUF1 flag in the PWR_SR1 register. This bit is always read as 0.

5.5.8 PWR control register 5 (PWR_CR5)

This register is not reset when exiting Standby modes.

Access: three additional APB cycles are needed to write this register versus a standard APB write.

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPSEN	RFEOLEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SMPSEN**: SMPS step-down converter enable

This bit enables the SMPS step-down converter.

0: SMPS step-down converter SMPS mode disabled (LDO mode enabled)

1: SMPS step-down converter SMPS mode enabled

Caution: Before enabling the SMPS, the SMPS clock detection must be enabled in SUBGHZ_SMPSC0R.CLKDE, if the application uses an external HSE clock source (not coming from XO or TCXO but from another device).

Bit 14 **RFEOLEN**: sub-GHz radio end-of-life detector enable

0: Radio end-of-life detector disabled

1: Radio end-of-life detector enabled

Bits 13:0 Reserved, must be kept at reset value.

5.5.9 PWR port A pull-up control register (PWR_PUCRA)

This register is not reset when exiting Standby modes.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port PA[y] pull-up bit y (y = 0 to 15)
When set, each bit activates the pull-up on PA[y] when both APC bits are set in *PWR control register 3 (PWR_CR3)*. The pull-up is not activated if the corresponding PA[y] bit is also set.

5.5.10 PWR port A pull-down control register (PWR_PDCRA)

This register is not reset when exiting Standby modes.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port PA[y] pull-down (y = 0 to 15)
When set, each bit activates the pull-down on PA[y] when both APC bits are set in *PWR control register 3 (PWR_CR3)*.

5.5.11 PWR port B pull-up control register (PWR_PUCRB)

This register is not reset when exiting Standby modes.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port PB[y] pull-up (y = 0 to 15)
When set, each bit activates the pull-up on PB[y] when both APC bits are set in *PWR control register 3 (PWR_CR3)*. The pull-up is not activated if the corresponding PB[y] bit is also set.

5.5.12 PWR port B pull-down control register (PWR_PDCRB)

This register is not reset when exiting Standby modes.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port PB[y] pull-down (y = 0 to 15)

When set, each bit activates the pull-down on PB[y] when both APC bits are set in *PWR control register 3 (PWR_CR3)*.

5.5.13 PWR port C pull-up control register (PWR_PUCRC)

This register is not reset when exiting Standby modes.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	Res.	Res.	Res.	Res.	Res.	Res.	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **PU[15:13]**: Port PC[y] pull-up (y = 13 to 15)

When set, each bit activates the pull-up on PC[y] when both APC bits are set in *PWR control register 3 (PWR_CR3)*. The pull-up is not activated if the corresponding PC[y] bit is also set.

Bits 12:7 Reserved, must be kept at reset value.

Bits 6:0 **PU[6:0]**: Port PC[y] pull-up (y = 0 to 6)

When set, each bit activates the pull-up on PC[y] when both APC bits are set in *PWR control register 3 (PWR_CR3)*. The pull-up is not activated if the corresponding PC[y] bit is also set.

5.5.14 PWR port C pull-down control register (PWR_PDCRC)

This register is not reset when exiting Standby modes.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	Res.	Res.	Res.	Res.	Res.	Res.	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **PD[15:13]**: Port PC[y] pull-down (y = 13 to 15)
When set, each bit activates the pull-down on PC[y] when both APC bits are set in *PWR control register 3 (PWR_CR3)*.

Bits 12:7 Reserved, must be kept at reset value.

Bits 6:0 **PD[6:0]**: Port PC[y] pull-down bit y (y = 0 to 6)
When set, each bit activates the pull-down on PC[y] when both APC bits are set in .

5.5.15 PWR port H pull-up control register (PWR_PUCRH)

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU3:** Port PH[3] pull-up
When set, this bit activates the pull-up on PH[3] when both APC bit are set in *PWR control register 3 (PWR_CR3)*. The pull-up is not activated if the corresponding PH[3] bit is also set.

Bits 2:0 Reserved, must be kept at reset value.

5.5.16 PWR port H pull-down control register (PWR_PDCRH)

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Access: additional APB cycles are needed to access this register versus those needed for a standard APB access (three for a write and two for a read).

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PD3:** Port PH[3] pull-down

When set, this bit activates the pull-down on PH[3] when both APC bits are set in *PWR control register 3 (PWR_CR3)*.

Bits 2:0 Reserved, must be kept at reset value.

5.5.17 PWR extended status and status clear register (PWR_EXTSCR)

Access: three additional APB cycles are needed to write this register versus a standard APB write.

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	C1DS	Res.	Res.	Res.	C1STOPF	C1STOP2F	C1SBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1CSSF
	r				r	r	r								w

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **C1DS**: CPU Deep-Sleep mode

This bit is set by hardware when CPU enters Deep-Sleep mode.
0: CPU is running or in sleep
1: CPU is in Deep-Sleep

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **C1STOPF**: System Stop 0, 1 flag for CPU (all core states retained)
This bit is set by hardware and cleared only by any reset or by setting C1CSSF bit.
0: System has not been in Stop 0 or 1 mode
1: System has been in Stop 0 or 1 mode.

Bit 9 **C1STOP2F**: System Stop 2 flag for CPU (partial core states retained)
This bit is set by hardware and cleared only by any reset or by setting C1CSSF bit.
0: System has not been in Stop 2 mode
1: System has been in Stop 2 mode.

Bit 8 **C1SBF**: System Standby flag for CPU (no core states retained)
This bit is set by hardware and cleared only by a POR reset or by setting C1CSSF bit.
0: System has not been in Standby mode
1: System has been in Standby mode.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **C1CSSF**: Clear CPU Stop Standby flags
Setting this bit clears the C1STOPF and C1SBF bits.

5.5.18 PWR sub-GHz SPI control register (PWR_SUBGHZSPICR)

Address offset: 0x090

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NSS**: sub-GHz SPI NSS control
This bit is set and cleared by software and is used to control the sub-GHz SPI NSS level from software.
0: sub-GHz SPI NSS signal at level low
1: sub-GHz SPI NSS signal is at level high

Bits 14:0 Reserved, must be kept at reset value.

5.5.19 PWR register map

Table 48. PWR register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPR	Res.	Res.	Res.	VOS [1:0]		DBP	Res.	Res.	FPDS	FPDR	SUBGHZSPINSSSEL	LPMS [2:0]		
	Reset value																		0				0	1	0			0	0	0	0	0	0
0x004	PWR_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVME3	Res.	Res.	PLS [2:0]			PVDE
	Reset value																										0			0	0	0	0
0x008	PWR_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EIWUL	Res.	EWRFIRQ	Res.	EWRFBUSY	APC	RRS	EWPVD	ULPEN	Res.	Res.	Res.	Res.	EWUP3	EWUP2	EWUP1
	Reset value																	1		0		0	0	0	0	0					0	0	0
0x00C	PWR_CR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRFBUSYP	Res.	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	WP3	WP2	WP1
	Reset value																					0		0	0						0	0	0
0x010	PWR_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFI	Res.	Res.	Res.	WRFBUSYF	Res.	Res.	WPVDF	Res.	Res.	Res.	Res.	Res.	WUF3	WUF2	WUF1
	Reset value																	0				0			0						0	0	0
0x014	PWR_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVMO3	Res.	Res.	PVDO	VOSF	REGLPF	REGLPS	FLASHRDY	REGMRS	RFEOLF	LDORDY	SMPSRDY	RFBUSYMS	RFBUSYS	Res.
	Reset value																		0			0	0	0	0	0	0	0	0	0	0	0	
0x018	PWR_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWRFBUSYF	Res.	Res.	CWPVDF	Res.	Res.	Res.	Res.	Res.	CWUF3	CWUF2	CWUF1
	Reset value																					0			0						0	0	0
0x01C	PWR_CR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPSEN	RFEOLEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0														
0x020	PWR_PUCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	PWR_PDCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	PWR_PUCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 48. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	PWR_PDCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	PWR_PUCRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU15	PU14	PU13	Res.	Res.	Res.	Res.	Res.	Res.	PU6	PU5	PU4	PU3	PU2	PU1	PU0
	Reset value																	0	0	0							0	0	0	0	0	0	0
0x034	PWR_PDCRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD15	PD14	PD13	Res.	Res.	Res.	Res.	Res.	Res.	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																	0	0	0							0	0	0	0	0	0	0
0x058	PWR_PUCRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU3	Res.	Res.	Res.
	Reset value																													0			
0x05C	PWR_PDCRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD3	Res.	Res.	Res.
	Reset value																													0			
0x088	PWR_EXTSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1DS	Res.	Res.	Res.	C1STOPF	C1STOP2F	C1SBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1CSSF
	Reset value																		0				0	0	0								0
0x090	PWR_ SUBGHZSPICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	1															

Refer to *Section 2.4* for the register boundary addresses.

6 Reset and clock control (RCC)

6.1 Reset

There are three types of reset, defined as system reset, power reset and Backup domain reset.

6.1.1 Power reset

A power reset is generated when one of the following events occurs:

- a Brownout reset (BOR)
- when exiting from Standby mode
- when exiting from Shutdown mode

A Brownout reset, including power-on or power-down reset (POR/PDR), sets all registers to their reset values except the Backup domain.

When exiting Standby mode, all registers in the V_{CORE} domain are set to their reset value. Registers outside the V_{CORE} domain (RTC, WKUP, IWDG and Standby/Shutdown modes control) are not impacted.

When exiting Shutdown mode, a Brownout reset is generated, resetting all registers except those in the Backup domain.

6.1.2 System reset

A system reset sets all registers to their reset values unless otherwise specified in the register description.

A system reset is generated when one of the following events occurs:

- a low level on the NRST pin (external reset)
- window watchdog event (WWDG reset)
- independent watchdog event (IWDG reset)
- a software reset (SW reset) (see *Software reset*)
- low-power mode security reset (see *Low-power mode security reset*)
- option byte loader reset (see *Option byte loader reset*)
- a Brownout reset
- an illegal sub-GHz radio access (sub-GHz radio protocol error reset) (only valid for non LoRa devices, STM32WLE4xx)

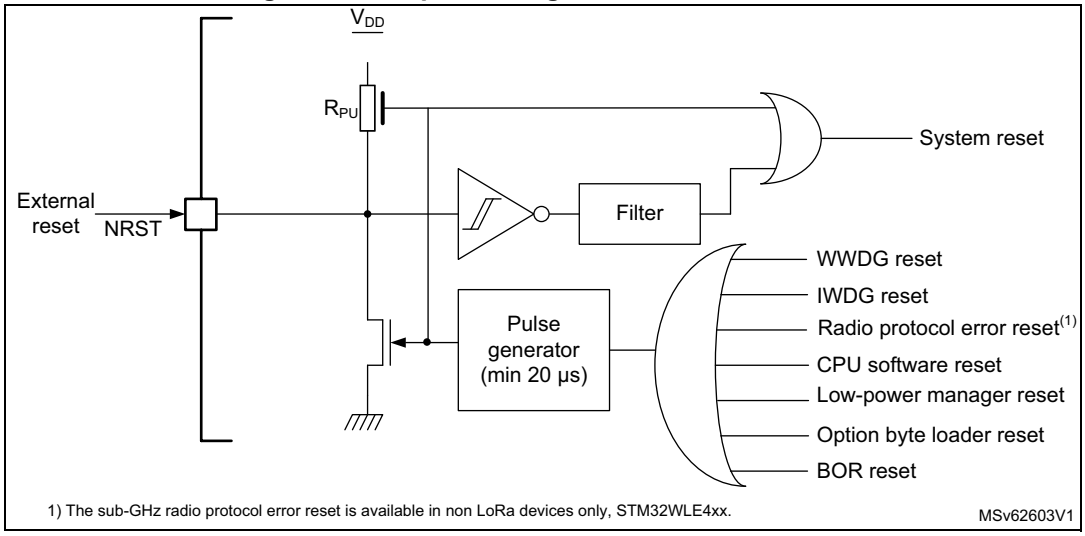
The reset source can be identified by checking the reset flags in the control/status register, RCC_CSR (see *Section 6.4.31: RCC control/status register (RCC_CSR)*).

These sources act on the NRST pin, that is always kept low during the delay phase. The CPU RESET service routine vector is selected via the BOOT0 and BOOT1.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

In case on an internal reset, the internal pull-up R_{PU} is deactivated in order to save the power consumption through the pull-up resistor.

Figure 20. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in CPU application interrupt and reset control register may be set to force a software reset on the device (refer to the programming manual *STM32 Cortex®-M4 MCUs and MPUs* (PM0214)).

Low-power mode security reset

To prevent that critical applications mistakenly enter a low-power mode, two low-power mode security resets are available.

If enabled in option bytes, the resets are generated in the following conditions:

- Entering Standby mode: this type of reset is enabled by resetting nRST_STDBY bit in user option bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
- Entering Stop mode: this type of reset is enabled by resetting nRST_STOP bit in user option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.
- Entering Shutdown mode: this type of reset is enabled by resetting nRST_SHDW bit in user option bytes. In this case, whenever a Shutdown mode entry sequence is successfully executed, the device is reset instead of entering Shutdown mode.

For further information on the user option bytes, refer to *Section 3.4.1: Option bytes description*.

Option byte loader reset

The option byte loader reset is generated when the OBL_LAUNCH bit is set in the FLASH_CR register. This bit is used to launch the option byte loading by software.

6.1.3 Backup domain reset

The Backup domain has two specific resets.

A Backup domain reset is generated when one of the following events occurs:

- a software reset, triggered by setting the BDRST bit in the *RCC Backup domain control register (RCC_BDCR)*
- V_{DD} or V_{BAT} power on, if both supplies have previously been powered off

A Backup domain reset only affects the LSE oscillator, the RTC, the Backup registers and the RCC Backup domain control register.

6.1.4 Sub-GHz radio reset

The sub-GHz radio can be reset by the RFRST register bit. A sub-GHz radio reset status flag is provided in the RFRSTF register bit. The sub-GHz radio must not be accessed as long as the reset status flag RFRSTF indicates sub-GHz radio in reset.

The sub-GHz radio is also reset when entering Shutdown mode.

6.1.5 PKA SRAM reset

The PKA SRAM is erased by hardware on any power reset and system reset. The status of the PKA SRAM erase operation can be monitored in SYSCFG_SCSR.PKASRAMBSY flag register bit.

6.2 Clocks

The following different clock sources can be used to drive the system clock (SYSCLK):

- HSI16 (high-speed internal) 16 MHz RC oscillator clock
- MSI (multi-speed internal) RC oscillator clock from 100 kHz to 48 MHz
- HSE32 (high-speed external) 32 MHz oscillator clock, with trimming capacitors.
- PLL clock

The MSI is used as system clock source after startup from reset, configured at 4 MHz.

The devices have the following additional clock sources:

- LSI: 32 kHz low-speed internal RC that may drive the independent watchdog and optionally the RTC used for auto-wakeup from Stop and Standby modes.
- LSE: 32.768 kHz low-speed external crystal that optionally drives the RTC used for auto-wakeup from Stop, Standby and Shutdown modes, or the real-time clock (RTCCLK).

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Several prescalers can be used to configure the AHB frequencies (HCLK3/PCLK3, HCLK1), the high-speed APB2 (PCLK2) and the low-speed APB1 (PCLK1) domains. The maximum frequency of the AHB (HCLK3, HCLK1), the PCLK1 and the PCLK2 domains is 48 MHz.

Most peripheral clocks are derived from their bus clock (HCLK, PCLK) except the following:

- The clock used for true RNG, is derived (selected by software) from one of the following sources:
  - PLL VCO (PLLQCLK) (only available in Run mode)
  - MSI (only available in Run mode)
  - LSI clock
  - LSE clock
- The ADC clock is derived (selected by software) from one of the following sources:
  - system clock (SYSCLK) (only available in Run mode)
  - HSI16 clock (only available in Run mode)
  - PLL VCO (PLLPCLK) (only available in Run mode)
- The DAC uses the LSI clock in sample and hold mode
- The (LP)U(S)ARTs clocks are derived (selected by software) from one of the following sources:
  - system clock (SYSCLK) (only available in Run mode)
  - HSI16 clock (available in Run and Stop modes)
  - LSE clock (available in Run and Stop modes)
  - APB clock (PCLK depending on which APB the U(S)ART is mapped) (available in CRun and CSleep when also enabled in (LP)U(S)ARTxSMEN.)

  The wakeup from Stop mode is supported only when the clock is HSI16 or LSE.
- The I2Cs clocks are derived (selected by software) from one of the following sources:
  - system clock (SYSCLK) (only available in Run mode)
  - HSI16 clock (available in Run and Stop modes)
  - APB clock (PCLK depending on which APB the I2C is mapped) (available in CRun and CSleep when also enabled in I2CxSMEN.)

  The wakeup from Stop mode is supported only when the clock is HSI16.
- The SPI2S2 I2S clock is derived (selected by software) from one of the following sources:
  - HSI16 clock (only available in Run mode)
  - PLL VCO (PLLQCLK) (only available in Run mode)
  - external input I2S_CK (available in Run and Stop modes)
- The low-power timers (LPTIMx) clock is derived (selected by software) from one of the following sources:
  - LSI clock (available in Run and Stop modes)
  - LSE clock (available in Run and Stop modes)
  - HSI16 clock (only available in Run mode)
  - APB clock (PCLK depending on which APB the LPTIMx is mapped) (available in Run and CStop when enabled in LPTIMxSMEN.)
  - external clock mapped on LPTIMx_IN1 (available in Run and Stop modes)

  The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE, or in external clock mode.
- The RTC clock is derived (selected by software) from one of the following sources:
  - LSE clock

– LSI clock
– HSE32 clock divided by 32

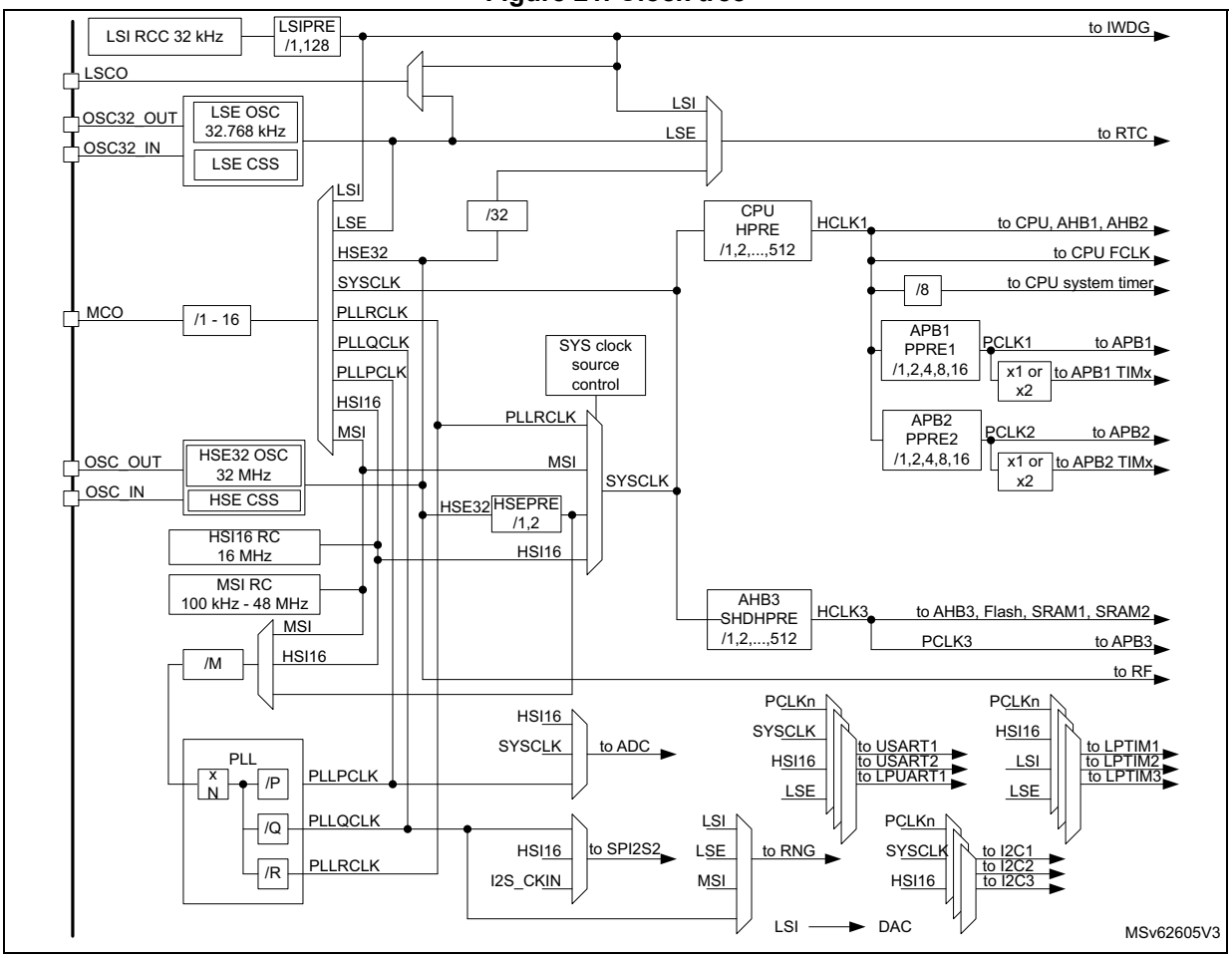
The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE.

- The IWDG clock is always the LSI clock.

The RCC feeds the CPU system timer (SysTick) external clock with the AHB clock (HCLK1) divided by eight. The SysTick can work either with this clock or directly with the CPU clock (HCLK1), configurable in the SysTick control and status register.

FCLK1 acts as CPU free-running clock. For more details, refer to the programming manual *STM32 Cortex®-M4 MCUs and MPUs* programming manual (PM0214).

Figure 21. Clock tree



1. For full details about the internal and external clock source characteristics, refer to the electrical characteristics section in the device datasheet.
2. The ADC clock can additionally be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is 1, the AHB prescaler must be equal to 1.

6.2.1 HSE32 clock with trimming

The HSE32 32 MHz external oscillator has the advantage of producing a very accurate rate on the main clock. The HSE32 furthermore provides on-chip trimming capability.

The high-speed external clock signal (HSE32) can be generated from the following clock sources:

- HSE32 external crystal
- HSE32 external clock
  - external clock source
  - external TCXO

The clock source must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time.

HSE32 is controlled from the CPU and from the sub-GHz radio (see *Section 4: Sub-GHz radio (SUBGHZ)*).

HSE32 can be switched on and off using the HSEON bit in the *RCC clock control register (RCC_CR)*. The HSE32 clock sources can be either an external crystal (XTAL) or an external source including a temperature compensated crystal oscillator (TCXO). HSE32 must be enabled with the HSEON bit when used for the CPU.

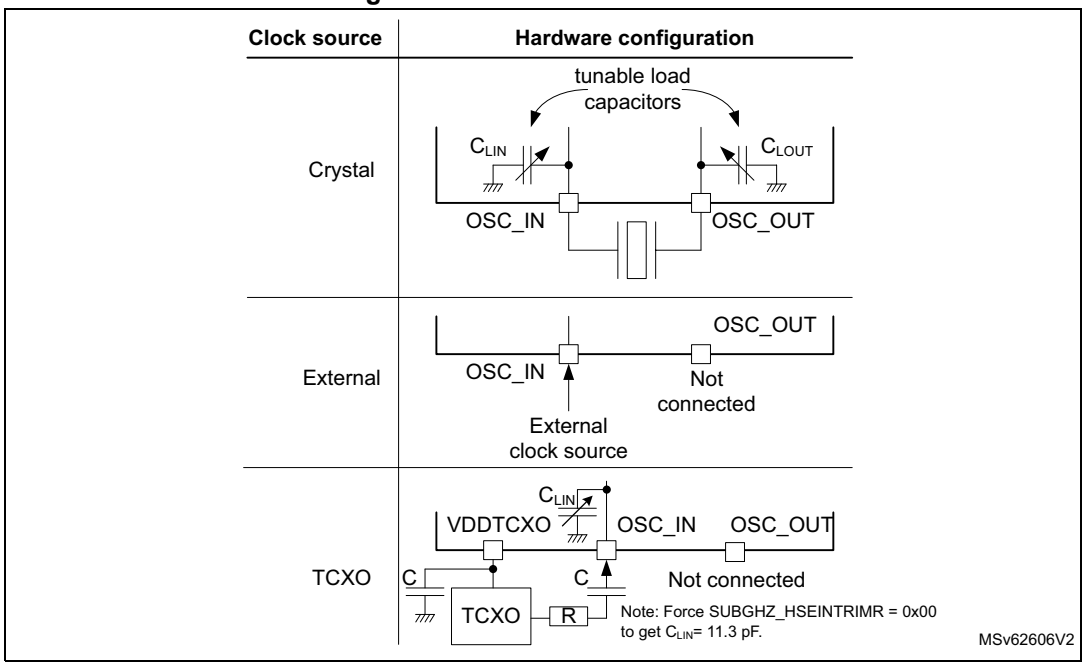
The stability of the XTAL HSE32 clock may be impacted by the sub-GHz radio, depending on the transmit output power (max +22 dBm), heating up the device. Heating depends on the used transmit output power and the device package. Careful PCB design using thermal heat dissipation techniques must be applied to avoid heat transfer to the HSE32 reference clock source. For the HSE32 frequency drift requirements related to the sub-GHz radio, see *Section 4.5.1: LoRa modem*.

The HSERDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSE32 oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

The sub-GHz radio enables HSE32 autonomously for its own purpose, independently of the HSEON bit.

Warning: The HSE32 cannot be used in LPRun mode.

Figure 22. HSE32 clock sources



External crystal (HSE32 crystal)

The associated hardware configuration is shown in *Figure 22: HSE32 clock sources*. Refer to the electrical characteristics section of the datasheet for more details.

Frequency trimming

When using HSE32 with external crystal, the load capacitors are provided by the integrated capacitor banks, that can be trimmed. The HSE32 load capacitor trimming allows a compensation of device manufacturing process variations, used crystal and PCB design. The HSE32 frequency can be tuned in the application via the sub-GHz radio registers SUBGHZ_HSEINTRIMR and SUBGHZ_HSEOUTRIMR. For more information see *Section 4.4: Sub-GHz radio clocks*.

The HSE32 frequency can be measured by outputting the HSE32 clock on the MCO when in Run mode.

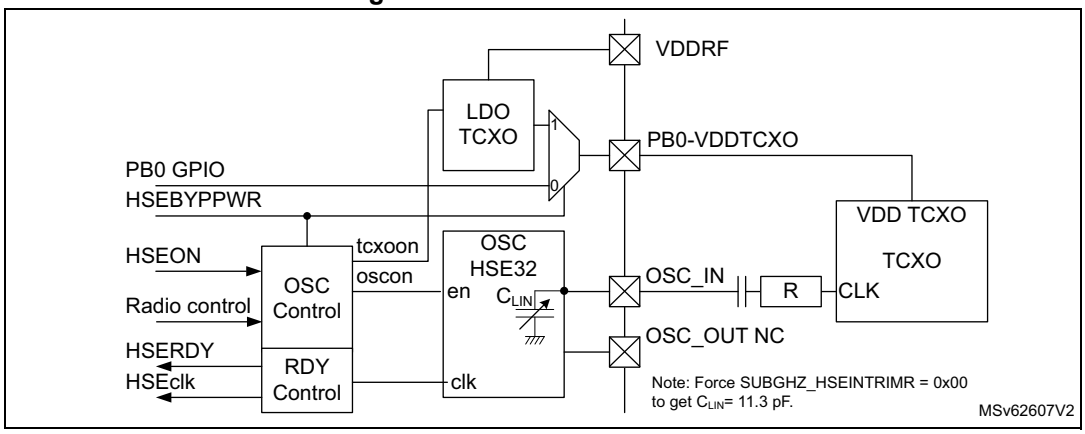
External source (HSE32 TXCO)

In this mode, an external TXCO clock source must be provided. This external source frequency must be 32 MHz. This mode is selected by setting the HSEBYPPWR and HSEON bits in the *RCC clock control register (RCC_CR)*. The external clock signal (refer to the datasheet) must drive the following pins (see *Figure 22: HSE32 clock sources*):

- OSC_IN and OSC_OUT pins: OSC_IN pin must be driven, while the OSC_OUT pin must be left not connected.

The TCXO supply can be provided by the device on PB0/VDDTCXO. The V_{DDTCXO} supply is also enabled with the HSEBYPPWR bit in *RCC clock control register (RCC_CR)* before enabling the HSE32 oscillator. V_{DDTCXO} supply level and TCXO clock startup timeout can be configured through the sub-GHz radio `Set_TcxoMode()` command (see *Section 4: Sub-GHz radio (SUBGHZ)* for more details).

Figure 23. HSE32 TCXO control



The control of the HSE32 external TCXO can be done in the following ways:

- powered by V_{DDTCXO}:
  - Set the voltage level inside the sub-GHz radio by using the `Set_TcxoMode()` command, before enabling the VDDTCXO regulator.
  - Set the HSEBYPPWR bit to 1 to use the VDDTCXO regulator.
  - Enable the HSE32 use with external TCXO in Run mode for the CPU with the HSEON bit.The availability of the clock can be checked with the HSERDY bit.
  - Enable the HSE32 use with TXCXO for the sub-GHz radio with the sub-GHz radio `Set_TcxoMode()` command when in sub-GHz radio Standby mode. It is recommended to issue a sub-GHz radio calibration command once the HSE32 clock is available.

When the CPUis in one of the low-power modes (Stop, Standby, or Shutdown) and the sub-GHz radio is in Sleep, the HSE32 clock including the TCXO is disabled.

6.2.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz Oscillator.

The HSI16 oscillator has the advantage of providing a clock source at low cost. It also has a faster startup time than the HSE32 crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

The HSI16 clock can be selected as system clock after wakeup from Stop modes (Stop 0, Stop 1 or Stop 2). Refer to *Section 6.3: Low-power modes*. It can also be used as a backup clock source (auxiliary clock) if the HSE32 crystal oscillator fails. Refer to *Section 6.2.10: Clock security system on HSE32 (CSS)*.

Calibration

The RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. This is why each device is factory calibrated by ST for 1 % accuracy at T_A = 25 °C.

After a reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the *RCC internal clock sources calibration register (RCC_ICSCR)*.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. The HSI16 frequency can be trimmed in the application using the HSITRIM[6:0] bits in the *RCC internal clock sources calibration register (RCC_ICSCR)*.

For more details on how to measure the HSI16 frequency variation, refer to *Section 6.2.20: Internal/external clock measurement with TIM16/TIM17*.

The HSIRDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSI16 RC is stable or not. At startup, the HSI16 RC output clock is not released until this bit is set by hardware.

The HSI16 RC can be switched on and off using the HSION bit in the *RCC clock control register (RCC_CR)*.

The HSI16 signal can also be used as a backup source (Auxiliary clock) if the HSE32 crystal oscillator fails. Refer to *Section 6.2.10: Clock security system on HSE32 (CSS) on page 242*.

6.2.3 MSI clock

The MSI clock signal is generated from an internal RC oscillator. Its frequency range can be adjusted by software using the MSIRANGE[3:0] bits in the *RCC clock control register (RCC_CR)*. The following frequency ranges are available: 100 kHz, 200 kHz, 400 kHz, 800 kHz, 1 MHz, 2 MHz, 4 MHz (default value), 8 MHz, 16 MHz, 24 MHz, 32 MHz and 48 MHz. To use the MSI range, it must be selected by MSIRGSEL.

The MSI clock is used as system clock after restart from Reset, wakeup from Standby and Shutdown low-power modes. After restart from Reset and Shutdown, the MSI frequency is set to its default value 4 MHz (see *Section 6.3: Low-power modes*). When wakeup from Standby, MSI can be adjusted by software using the MSISRANGE[3:0] bits in the *RCC control/status register (RCC_CSR)*. The following frequency ranges are available: 1MHz, 2 MHz, 4 MHz (default value) and 8 MHz.

The MSI clock can be selected as system clock after a wakeup from Stop mode (Stop 0, Stop 1 or Stop 2, see *Section 6.3: Low-power modes*). It can also be used as a backup clock source (auxiliary clock for the CPU) if the HSE32 crystal oscillator fails (see *Section 6.2.10: Clock security system on HSE32 (CSS)*).

The MSI RC oscillator provides a low-power clock source. In addition, when used in PLL-mode with the LSE, it also provides a very accurate clock source that can be used to feed the PLL to run the system at the maximum speed 48 MHz.

The MSIRDY flag in the *RCC clock control register (RCC_CR)* indicates whether the MSI RC is stable or not. At startup, the MSI RC output clock is not released until this bit is set by hardware. The MSI RC can be switched on and off by using the MSION bit in the *RCC clock control register (RCC_CR)*.

Hardware auto calibration with LSE (PLL-mode)

When a 32.768 kHz external oscillator is present in the application, it is possible to configure the MSI in a PLL-mode by setting the MSIPLLEN bit in the *RCC clock control register (RCC_CR)*. When configured in PLL-mode, MSI automatically calibrates itself thanks to the LSE. This mode is available for all MSI frequency ranges.

Software calibration

The MSI RC oscillator frequency can vary from one chip to another due to manufacturing process variations. This is why each device is factory calibrated by ST for 1 % accuracy at an ambient temperature T_A = 25 °C. After reset, the factory calibration value is loaded in the MSICAL[7:0] bits in the *RCC internal clock sources calibration register (RCC_ICSCR)*. If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. The MSI frequency can be trimmed in the application using the MSITRIM[7:0] bits in the *RCC internal clock sources calibration register (RCC_ICSCR)*. For more details on how to measure the MSI frequency variation, refer to *Section 6.2.20: Internal/external clock measurement with TIM16/TIM17*.

6.2.4 PLL

The device embeds one PLL. The PLL provides up to three independent outputs. The internal PLL can be used to multiply the HSI16, HSE32 or MSI output clock frequency. The PLL input frequency must be between 2.66 and 16 MHz. The selected clock source is divided by a programmable factor PLLM from 1 to 8 to provide a clock frequency in the requested input range. Refer to *Figure 21: Clock tree* and *RCC PLL configuration register (RCC_PLLCFGR)*.

The PLL configuration (selection of the input clock and multiplication factor) must be done before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLON to 0 in the *RCC clock control register (RCC_CR)*.
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, PLLREN in the *RCC PLL configuration register (RCC_PLLCFGR)*.

An interrupt can be generated when the PLL is ready, if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

The PLLQCLK and PLLRCLK output frequency must not exceed 48 MHz. The PLLPCLK output frequency must not exceed 62 MHz.

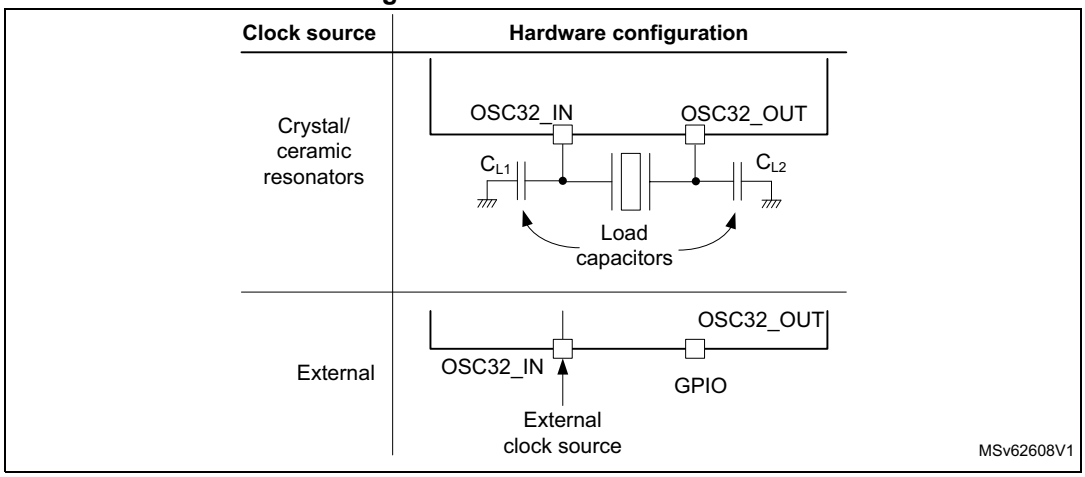
The enable bits of the PLL output clock (PLLPEN, PLLQEN, PLLREN) can be modified at any time without stopping the PLL. PLLREN cannot be cleared if PLLRCLK is used as system clock.

6.2.5 LSE clock

The LSE crystal is a 32.768 kHz low-speed external crystal or ceramic resonator. It provides a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

The resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

Figure 24. LSE clock sources



The LSE crystal is switched on and off using the LSEON bit in the *RCC Backup domain control register (RCC_BDCR)*. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the *RCC Backup domain control register (RCC_BDCR)* to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other side. The LSE drive can be decreased to the lower drive capability (LSEDRV = 0) when the LSE is on. However, once LSEDRV is selected, the drive capability can not be increased if LSEON = 1.

The LSERDY flag in the *RCC Backup domain control register (RCC_BDCR)* indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

When enabled and ready the LSE clock can directly be used by the RTC. To be able to use the clocks by other peripherals (LPTIMx, TIMx, USARTx, LPUARTx, system LSCO, MCO, MSI PLL mode), the LSE system clock must be enabled with the LSESYSEN bit in the *RCC Backup domain control register (RCC_BDCR)*. When the LSE clock is ready and LSECSS is enabled, the LSE clock is used by the LSECSS and is available on the LSCO. A LSESYSRDY flag is provided in the *RCC Backup domain control register (RCC_BDCR)* to indicate when LSE system clock is ready (due clock synchronization) after having been enabled by the LSESYSEN.

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. This mode is selected by setting the LSEBYP and LSEON bits in the *RCC Backup domain control register (RCC_BDCR)*. The external clock signal (square, sinus or triangle) with ~50 % duty cycle must drive the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO (see *Figure 24: LSE clock sources*).

6.2.6 LSI clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby modes for the independent watchdog (IWDG) and RTC. The clock frequency is ~32 kHz or can be divided by 128 (~250 Hz) using LSIPRE. For more details, refer to the electrical characteristics section of the datasheets.

The LSI RC can be switched on and off using the LSION bit in the *RCC control/status register (RCC_CSR)*.

The LSIRDY flag in the *RCC control/status register (RCC_CSR)* indicates if the LSI oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

6.2.7 Clock source stabilization time

The different clock sources require a stabilization time, during which no clock is forwarded to the system (see the table below).

Table 49. Clock source stabilization times

Clock source	Stabilization time
MSI	Refer to the device datasheet.
HSI	Refer to the device datasheet.
HSE	Refer to the device datasheet.
LSI	2 cycles (~85 µs LSIPRE = 0)
	2 cycles (~2 ms LSIPRE = 1)
LSE	4096 cycles (125 ms)

6.2.8 System clock (SYSCLK) selection

The following clock sources can be used to drive the system clock (SYSCLK):

- MSI oscillator
- HSI16 oscillator
- HSE32 oscillator (either 32 MHz or divided by 2 for 16 MHz)
- PLLRCLK

The system clock maximum frequency in range 1 is 48 MHz. After a system reset, the MSI oscillator, at 4 MHz, is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *RCC internal clock sources calibration register (RCC_ICSCR)* indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

When waking up from Standby mode, the MSI at 4 MHz is selected as system clock.

In range 2, the system clock must not exceed 16 MHz.

6.2.9 Clock source frequency versus voltage scaling

The following table gives the different clock source frequencies depending on the product voltage range.

Table 50. Clock source frequency

Product voltage range	Clock frequency			
	MSI	HSI16	HSE32	PLL
Range 1	48 MHz	16 MHz	32 MHz	PLLRCLK = PLLQCLK = 48 MHz PLLPCLK = 62 MHz (VCO max = 344 MHz)
Range 2	16 MHz	16 MHz	32 MHz(1)	PLLRCLK = PLLQCLK = 16 MHz PLLPCLK = 21 MHz (VCO max = 128 MHz)

1. The HSEPRE must be set to divide by two.

6.2.10 Clock security system on HSE32 (CSS)

The clock security system can be activated by software. In this case, the clock detector is enabled after the HSE32 oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE32 clock, the HSE32 oscillator is automatically disabled. A clock failure event is sent to the break input of the advanced-control timers (TIM1 and TIM16/17) and a HSE32 CSS interrupt is generated to inform the software about the failure, allowing the MCU to perform rescue operations. The HSE32 CSS interrupt is linked to the CPU NMI (non-maskable interrupt) exception vector.

Note: Once the HSE32 CSS is enabled and if the HSE32 clock fails, the HSE32 CSS interrupt *occurs and a NMI is automatically generated. The NMI is executed indefinitely unless the* CSSF *pending bit is cleared. As a consequence, in the NMI ISR (interrupt service routine), the user must clear the* HSE CSS interrupt *by setting the* CSSC *bit in the RCC clock interrupt clear register (RCC_CICR).*

If the HSE32 oscillator is used directly or indirectly as the system clock (indirectly meaning HSE32 is used as PLL input clock and the PLL clock is used as system clock), a detected failure causes a switch of the system clock to the MSI or the HSI16 oscillator depending on the STOPWUCK configuration in the *RCC clock configuration register (RCC_CFGR)*, and the disabling of the HSE32 oscillator. If the HSE32 clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

6.2.11 Clock security system on LSE (LSECSS)

A CSS on LSE can be activated by software writing the LSECSSON bit in the *RCC Backup domain control register (RCC_BDCR)*. This bit can be disabled only by a hardware reset or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE and LSI are enabled (LSEON and LSION) and ready (LSERDY and LSIRDY set by hardware), and after the RTC clock has been selected by RTCSEL. The LSI clock is automatically enabled.The LSE must not be disabled with the LSEON bit when LSECSS is enabled with the LSECSSON bit.

The CSS on LSE is working in all modes except VBAT. It is working also under system reset (excluding power on reset). If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC but no hardware action is made to the registers. If the MSI was in PLL-mode, this mode is disabled.

In Standby mode, a wakeup is generated. In other modes an interrupt can be sent to wakeup the software (see *RCC clock interrupt enable register (RCC_CIER)*, *RCC clock interrupt flag register (RCC_CIFR)*, *RCC clock interrupt clear register (RCC_CICR)*).

The software must then disable the LSECSSON bit, stop the defective 32 kHz oscillator (disabling LSEON), and change the RTC clock source (no clock or LSI or HSE32, with RTCSEL), or take any required action to secure the application.

6.2.12 SPI2S2 clock

The SPI2S2 I2S clock is derived from the HSI16 clock, PLL output, or from the external I2S_CLK signal. It can reach 62 MHz.

The serial audio interface requires either a frequency close to 49.152 MHz or 11.2896 MHz. The 49.152 MHz aims to derive audio sampling frequencies of 192 kHz, 96 kHz, 48 kHz, 32 kHz, 16 kHz and 8 kHz. While 11.2896 MHz targets audio sampling frequencies of 44.1 kHz, 22.05 kHz and 11.025 kHz. The targeted worts case accuracy must be 0.05 %.

Possible clock configurations are given in the table below.

Table 51. SPI2S2 I2S clock PLL configurations

Clock source	M	PLLN	PLLQ	I2C clock frequency
MSI (4 MHz)	1	4	7	49.14286 MHz (-0.019%)
HSE32 (32 MHz)	7	43	4	
HSI16 (16 MHz)	2	43	7	
MSI (4 MHz)	1	79	28	11.28571 MHz (0.034%)
HSE32 (32 MHz)	3	18	17	11.29412 MHz (0.040%)
HSI16 (16 MHz)	1	12	17	

6.2.13 Sub-GHz radio SPI clock

The sub-GHz radio SPI clock is derived from the PCLK3 clock. The SUBGHZSPI_SCK frequency is obtained by PCLK3 divided by two. The SUBGHZSPI_SCK clock maximum speed must not exceed 16 MHz.

Table 52. Sub-GHz radio SPI clock configurations

PCLK3 [MHz]	SUBGHZSPI_SCK clock maximum speed
48	PCLK / 4(1) = 12 MHz
32	PCLK / 2(1) = 16 MHz

1. As controlled by SUBGHZSPI_CR1 BR baud rate control.

6.2.14 ADC clock

The ADC clock is derived from the system clock, from the HSI16 clock, or from the PLL output. The ADC clock can reach 35 MHz and can be divided by the following prescalers values: 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128 or 256 by configuring the ADC_CCR register. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bit fields in the ADC_CCR register.

If the programmed factor is 1, the AHB prescaler must be set to 1.

6.2.15 RTC clock

The RTCCLK clock source can be either the HSE32 divided by 32, the LSE or the LSI clock. RTCCLK is selected by programming the RTCSEL[1:0] bits in the *RCC Backup domain control register (RCC_BDCR)*. This selection cannot be modified without resetting the Backup domain. The system must always be configured so as to get a PCLK frequency greater then or equal to the RTCCLK frequency for a proper operation of the RTC.

The LSE clock is in the Backup domain, whereas the HSE32 and LSI clocks are not, with the following consequences:

- If LSE is selected as RTC clock, the RTC continues to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If LSI is selected as the RTC clock, the RTC state is not guaranteed if the V_{DD} supply is powered off.
- If the HSE32 clock divided by a prescaler is used as the RTC clock, the RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the V_{CORE} domain).

When the RTC clock is LSE or LSI, the RTC remains clocked and functional under system reset.

6.2.16 Timer clock

The timer clock frequencies are automatically defined by hardware.

The following cases are possible:

- If the APB prescaler (PPREx) selects the PCLKx clock to be HCLK1 not divided, the timer clock frequencies are set to the HCLK1 frequency (timer clock = HCLK1).
- If the APB prescaler (PPREx) selects the PCLKx clock to be HCLK1 divided by n, the timer clock frequencies are set to HCLK1 divided by (n / 2) (timer clock = 2 x PCLKx).

6.2.17 Watchdog clock

If the independent watchdog (IWDG) is started by an hardware option or a software access, the LSI clock is forced on.

If the LSI oscillator is disabled when starting the IWDG, the LSI oscillator is forced on. After the LSI oscillator temporization, the clock is provided to the IWDG.

6.2.18 True RNG clock

The true random number generator (RNG) seed clock is derived from the MSI, from the PLL output or from the LSE or LSI clock. It can reach 48 MHz and can be divided by a prescalers values by configuring the true RNG register. It is asynchronous to the AHB clock.

6.2.19 Clock-out capability

- MCO

  The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. One of the following clock signals can be selected as the MCO clock:

  – SYSCLK
  – MSI
  – HSI16 (only available when enabled by HSION)
  – HSE32
  – PLLRCLK
  – LSI
  – LSE
  – PLLPCLK
  – PLLQCLK

  The selection is controlled by the MCOSEL[3:0] bits in the *RCC clock configuration register (RCC_CFGR)*. The selected clock can be divided with the MCOPRE[2:0] bits in the *RCC clock configuration register (RCC_CFGR)*.

  The clock on MCO is only available in Run modes and is not available in Stop, Standby and Shutdown modes.

- LSCO

  Another output (LSCO) allows one of the following low-speed clocks to be output onto the external LSCO pin:

  – LSI
  – LSE

  The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN bit in the *RCC Backup domain control register (RCC_BDCR)*.

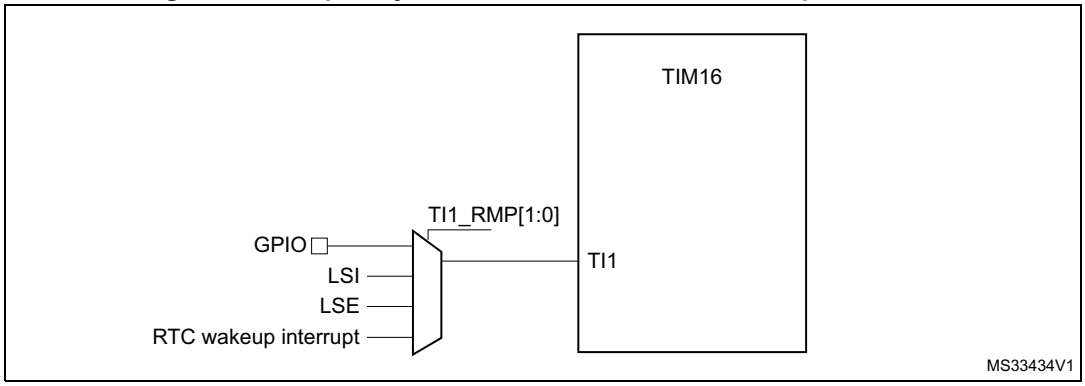
  The clock on LSCO is available in Run, Stop, and Standby and Shutdown modes.

The configuration registers of the corresponding GPIO port must be programmed in alternate function mode.

6.2.20 Internal/external clock measurement with TIM16/TIM17

The frequency of all on-board clock sources can be indirectly measured by mean of the TIM16 or TIM17 channel 1 input capture, as shown in *Figure 25* and *Figure 26*.

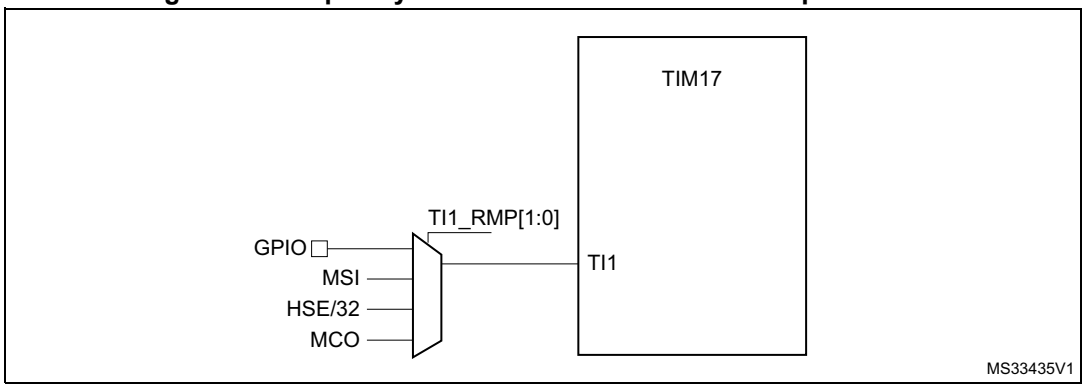
Figure 25. Frequency measurement with TIM16 in capture mode



The TIM16 input capture channel can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1_RMP[1:0] bits in the TIM16_OR register. The possibilities are listed below:

- TIM16 channel1 is connected to the GPIO (refer to the alternate function mapping in the device datasheets).
- TIM16 channel1 is connected to the LSI clock.
- TIM16 channel1 is connected to the LSE clock.
- TIM16 channel1 is connected to the RTC wakeup interrupt signal. In this case the RTC interrupt must be enabled.

Figure 26. Frequency measurement with TIM17 in capture mode



The TIM17 input capture channel can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1_RMP[1:0] bits in the TIM17_OR register. The possibilities are listed below:

- TIM17 channel1 is connected to the GPIO (refer to the alternate function mapping in the device datasheets).
- TIM17 channel1 is connected to the MSI Clock.
- TIM17 channel1 is connected to the HSE32/32 Clock.
- TIM17 channel1 is connected to the microcontroller clock output (MCO). This selection is controlled by the MCOSEL[3:0] bits in the *RCC clock configuration register (RCC_CFGR)*.

Calibration of the HSI16 and the MSI

For TIM16, the primary purpose of connecting the LSE to the channel 1 input capture is to be able to precisely measure the HSI16 and MSI system clocks (for this, HSI16 or MSI must be used as the system clock source). The number of HSI16 (MSI, respectively) clock counts between consecutive edges of the LSE signal, provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppms), it is possible to determine the internal clock frequency with the same resolution and to trim the source to compensate for manufacturing-process- and/or temperature- and voltage-related frequency deviations.

The MSI and HSI16 oscillator both have dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (the HSI16/LSE ratio): the precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement is.

If LSE is not available, HSE32/32 is the better option in order to reach the most precise calibration possible.

It is however not possible to have a good enough resolution when the MSI clock is low (typically below 1 MHz). In this case, the following is recommended:

- Accumulate the results of several captures in a row.
- Use the timer input capture prescaler (up to one capture every eight periods).
- Use the RTC wakeup interrupt signal (when the RTC is clocked by the LSE) as the input for the channel 1 input capture. This improves the measurement precision. For this purpose the RTC wakeup interrupt must be enabled.

Calibration of the LSI

The LSI calibration follows the same pattern than the HSI16, but changing the reference clock. It is necessary to connect the LSI clock to the TIM16 channel 1 input capture. Then the HSE32 must be defined as system clock source. The number of HSE32 clock counts between consecutive edges of the LSI signal provides a measure of the internal low-speed clock period.

The basic concept consists in providing a relative measurement (the HSE32 / LSI ratio): the precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement is.

6.2.21 Peripheral clocks enable

Most peripheral bus and kernel clocks can be individually enabled.The RCC_AHBxENR and RCC_APBxENRy registers enable peripheral clocks. The peripheral clocks follow the CPU state and the system state (see the table below). The RTC kernel clock is enabled by the RTCEN bit and does not depend on the CPU state nor the system state.

Peripheral bus clock activity during the CPU Sleep mode is controlled by the xxxxSMEN bit of the RCC_AHBxSMENR and RCC_APBxSMENRy registers.The peripheral bus clock during Sleep mode follows the CPU state (see the table below).

Table 53. Peripheral clock enable

xxxEN	xxxSMEN	System mode	Bus clock	Kernel clock(1)
0	x	Any	Stopped	Stopped
1		Run	Clocked	Clocked
	0	Sleep	Stopped	Clocked
	1	Sleep	Clocked	Clocked
		Stop	Stopped	Clocked when from HSI16, LSI, or LSE
				Stopped when from bus clock, SYSCLK, PLL clocks, or MSI clocks
x	x	Standby or Shutdown	Stopped	Stopped

1. Only the I2C, LPTIM, USART, LPUART, True RNG, ADC and SPI2S2 peripherals have a kernel clock controlled by xxxEN and xxxSMEN. The RTC has a kernel clock controlled by RTCEN and does not depend on xxxEN and xxxSMEN.

When the peripheral bus clock is not active, read or write accesses to the peripheral registers are not supported.

When the peripheral kernel clock is not active, the peripheral functionality is stopped.

The enable bits have a synchronization mechanism to create a glitch free clock for the peripheral. After the enable bit is set, there is a two clock cycles delay before the clock is active in the peripheral.

Caution: Just after enabling a clock for a peripheral, the software must wait for a delay before accessing the peripheral registers.

6.3 Low-power modes

AHB and APB peripheral clocks, including DMA clock, can be disabled by software.

Sleep and LPSleep modes stop the CPU clock. The memory interface clocks (Flash memory and SRAM1/2 interfaces) can be stopped during Sleep mode by software using the SRAMxSMEN bits. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled in the peripheral SMEN bits.

Stop modes (Stop 0, Stop 1 and Stop 2) stop most clocks in the V_{CORE} domain and disable the PLLs, the MSI and the HSE32 oscillators. HSI16 may be kept running when requested

by the peripheral (USART1, USART2, LPUART1, I2C1, I2C2 or I2C3) that allows the wakeup from Stop modes.

All U(S)ARTs, LPUARTs and I2Cs have the capability to enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as clock source for that peripheral).

All U(S)ARTs, LPUARTs and LPTIMs can also be driven by the LSE oscillator when the system is in Stop mode (if LSE is selected as clock source for that peripheral) and the LSE oscillator is enabled (LSEON). In that case, LSE remains always on in Stop mode (no capability to turn on the LSE oscillator).

All LPTIMs can also be driven by the LSI oscillator when the system is in Stop mode (if LSI is selected as clock source for that peripheral) and the LSI oscillator is enabled (LSION).

Standby and Shutdown modes stop all clocks in the V_{CORE} domain and disable the PLL, the HSI16, the MSI and the HSE32 oscillators.

The low-power modes can be overridden for debugging the CPU by setting the DBG_SLEEP, DBG_STOP or DBG_STANDBY bits in the DBGMCU_CR register. In addition, the EXTI CDBGPWRUPREQ events can be used to allow debugging the CPU in Stop modes (see the table below).

Table 54. Low-power debug configurations

<table>
<tr><th rowspan="2">Mode</th><th>CDBGPWRUPREQ</th><th colspan="3">DBGMCU</th><th>Debug</th></tr>
<tr><th>CPU</th><th>DBG_STANDBY</th><th>DBG_STOP</th><th>DBG_SLEEP</th><th>CPU</th></tr>
<tr><td>Sleep</td><td>x(1)</td><td>x</td><td>x</td><td>x</td><td>Enabled</td></tr>
<tr><td rowspan="2">Stop 0 and Stop 1</td><td>Disabled</td><td rowspan="3">x</td><td rowspan="2">Disabled</td><td rowspan="3">x</td><td>Disabled</td></tr>
<tr><td>Enabled</td><td>Enabled</td></tr>
<tr><td>Stop 0, Stop 1 and Stop 2</td><td>x</td><td>Enabled</td><td>Enabled</td></tr>
<tr><td rowspan="2">Standby</td><td rowspan="2">x</td><td>Disabled</td><td rowspan="2">x</td><td rowspan="2">x</td><td>Disabled</td></tr>
<tr><td>Enabled</td><td>Enabled</td></tr>
</table>

1. x = Don’t care.

When leaving the Stop modes (Stop 0, Stop 1 or Stop 2), the system clock is either MSI or HSI16, depending on the software configuration of the STOPWUCK bit in the *RCC clock configuration register (RCC_CFGR)*. The frequency (range and user trim) of the MSI oscillator is the one configured before entering Stop mode. The user trim of HSI16 is kept. If the MSI was in PLL-mode before entering Stop mode, the PLL-mode stabilization time must be waited for after wakeup, even if LSE was kept on during the Stop mode.

When exiting Standby mode, the system clock is MSI at 4 MHz.

When exiting Shutdown modes, the system clock is MSI. The MSI frequency at wakeup from Shutdown mode is 4 MHz. The user trim is lost.

If a Flash memory programming operation is ongoing, Stop, Standby and Shutdown modes entry is delayed until the Flash memory interface access is finished. If an access to the APB

domain is ongoing, Stop, Standby and Shutdown modes entry is delayed until the APB access is finished.

6.4 RCC registers

6.4.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0061

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PLL RDY	PLLON	Res.	Res.	HSEBY PPWR	HSE PRE	CSS ON	Res.	HSE RDY	HSEON
						r	rw			rw	rw	rs		r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	HSI KERDY	HSI ASFS	HSI RDY	HSI KERON	HSION	MSIRANGE[3:0]				MSIRG SEL	MSI PLLEN	MSI RDY	MSION
			r	rw	r	rw	rw	rw	rw	rw	rw	rs	rw	r	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **PLLRDY:** Main PLL clock ready flag

This bit is set by hardware to indicate that the main PLL is locked.

0: PLL unlocked

1: PLL locked

Bit 24 **PLLON:** Main PLL enable

This bit is set and cleared by software to enable the main PLL. It is also cleared by hardware when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the main PLL clock is used as the system clock.

0: Main PLL off

1: Main PLL on

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **HSEBYPPWR:** HSE32 VDDTCXO output on package pin PB0-VDDTCXO enable

This bit is set and cleared by software to control the function on package pin PB0-VDDTCXO. It can only be written when HSE32 oscillator is disabled (HSEON = HSERDY = 0).

0: PB0 selected

1: VDDTCXO selected

Bit 20 **HSEPRE:** HSE32 SYSCLK prescaler

This bit is set and cleared by software to control the division factor of SYSCLK when selecting HSE32 clock.

0: SYSCLK not divided (HSE32)

1: SYSCLK divided by two (HSE32 / 2)

Bit 19 **CSSON:** HSE32 clock security system enable

This bit is set by software to enable the clock security system. When CSSON is set, the HSE32 lock detector is enabled by hardware when the HSE32 oscillator is ready, and disabled by hardware if a HSE32 clock failure is detected. This bit is set only and is cleared by reset.

0: HSE32 CSS off (clock detector off)

1: HSE32 CSS on (clock detector on if the HSE32 oscillator is stable and off if not)

Bit 18 Reserved, must be kept at reset value.

Bit 17 **HSERDY:** HSE32 clock ready flag

This bit is set and cleared by hardware to indicate that the HSE32 oscillator is stable or not.

0: HSE32 oscillator not ready

1: HSE32 oscillator ready

Note: Once HSEON is cleared, HSERDY goes low after six HSE32 clock cycles.

Bit 16 **HSEON:** HSE32 clock enable for CPU

This bit is set and cleared by software. It is also cleared by hardware to stop the HSE32 oscillator when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the HSE32 oscillator is used directly or indirectly as system clock. The HSE32 oscillator must be disabled before entering LPRun mode.

0: HSE32 oscillator for CPU disabled

1: HSE32 oscillator for CPU enabled

Note: The sub-GHz radio has its own HSE32 oscillator enable in the sub-GHz radio.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **HSIKERDY:** HSI16 kernel clock ready flag for peripherals requests

This bit is set and cleared by hardware to indicate that the HSI16 oscillator is stable or not, when enabled by HSIKERON or a peripheral kernel clock request. This bit is not set when HSI16 is enabled by software with HSION setting or by wakeup from Standby.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

Note: Once HSIKERON is cleared, HSIKERDY goes low after six HSI16 clock cycles.

Bit 11 **HSIASFS:** HSI16 automatic start from Stop modes

This bit is set and cleared by software. When the system wakeup clock is MSI, this bit is used to wakeup the HSI16 in parallel to the system wakeup clock.

0: HSI16 not enabled by hardware when exiting Stop modes with MSI as wakeup clock

1: HSI16 enabled by hardware when exiting Stop mode with MSI as wakeup clock

Bit 10 **HSIRDY:** HSI16 clock ready flag

This bit is set and cleared by hardware to indicate that HSI16 oscillator is stable or not. It is set only when HSI16 is enabled by software by setting HSION, or by wakeup from Stop modes and HSIASFS is enabled. After wakeup from Stop modes, this bit is read 1 once the HSI16 is ready. This bit is not set when HSI16 is enabled by HSIKERON or by a peripheral request.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

Note: Once HSION is cleared, HSIRDY goes low after six HSI16 clock cycles.

Bit 9 **HSIKERON:** HSI16 enable for peripheral kernel clocks

This bit is set and cleared by software to force HSI16 on even in Stop modes. HSI16 enabled by HSIKERON can only feed USARTs, LPUARTs and I2Cs peripherals configured with HSI16 as kernel clock. Keeping HSI16 on in Stop modes avoids slowing down the communication speed because of the HSI16 startup time. This bit has no effect on HSION value.

0: No effect on HSI16 oscillator

1: HSI16 oscillator forced on even in Stop modes

Bit 8 **HSION:** HSI16 clock enable

This bit is set and cleared by software. It is also cleared by hardware to stop the HSI16 oscillator when entering Stop, Standby or Shutdown mode. This bit is set by hardware to force the HSI16 oscillator on when STOPWUCK = 1 or HSIASFS = 1 when exiting Stop modes, or in case of HSE32 crystal oscillator failure.

This bit is set by hardware if the HSI16 is used directly or indirectly as system clock. This bit cannot be reset if the HSI16 oscillator is used directly or indirectly as system clock.

0: HSI16 oscillator off

1: HSI16 oscillator on

Bits 7:4 **MSIRANGE[3:0]:** MSI clock ranges

These bits are configured by software to choose the frequency range of MSI when MSIRGSEL = 1. The following frequency ranges available:

0000: Range 0 around 100 kHz

0001: Range 1 around 200 kHz

0010: Range 2 around 400 kHz

0011: Range 3 around 800 kHz

0100: Range 4 around 1 MHz

0101: Range 5 around 2 MHz

0110: Range 6 around 4 MHz (reset value)

0111: Range 7 around 8 MHz

1000: Range 8 around 16 MHz

1001: Range 9 around 24 MHz

1010: Range 10 around 32 MHz

1011: Range 11 around 48 MHz

Others: not allowed (hardware write protection)

Caution: This field can be modified only when MSI is off (MSION = 0) or when MSI is ready (MSIRDY = 1). This filed **must not** be modified when MSI is on and when MSI is **not** ready (MSION = 1 and MSIRDY = 0).

Bit 3 **MSIRGSEL:** MSI range control selection

This bit is cleared to 0 on a system reset and when exiting Standby mode. It can be set to 1 by software. Software writing 0 has no effect.

0: MSI frequency range defined by MSISRANGE[3:0] in the RCC_CSR register

1: MSI frequency range defined by MSIRANGE[3:0] in the RCC_CR register

Bit 2 **MSIPLLEN:** MSI clock PLL enable

This bit is set and cleared by software to enable/disable the PLL part of the MSI clock source. It must be enabled after LSE is enabled (LSEON = 1) and ready (LSERDY set by hardware).There is a hardware protection to avoid enabling this bit if LSE is not ready.

This bit is cleared by hardware when LSE is disabled (LSEON = 0) or when the CSS on LSE detects an LSE failure (refer to RCC_CSR register).

0: MSI PLL off

1: MSI PLL on

Bit 1 **MSIRDY:** MSI clock ready flag

This bit is set and cleared by hardware to indicate that the MSI oscillator is stable or not. After reset, this bit is read 1 once the MSI is ready.

0: MSI oscillator not ready

1: MSI oscillator ready

Note: Once MSION is cleared, MSIRDY goes low after six MSI clock cycles.

Bit 0 **MSION:** MSI clock enable

This bit is set and cleared by software. It is also cleared by hardware to stop the MSI oscillator when entering Stop, Standby or Shutdown mode. This bit is set by hardware to force the MSI oscillator on when exiting Standby or Shutdown mode. It is set by hardware to force the MSI oscillator on when STOPWUCK = 0 when exiting from Stop modes, or in case of a HSE32 oscillator failure. This bit is set by hardware when used directly or indirectly as system clock. It cannot be reset if the MSI oscillator is used directly or indirectly as system clock.

0: MSI oscillator off

1: MSI oscillator on

6.4.2 RCC internal clock sources calibration register (RCC_ICSCR)

Address offset: 0x004

Reset value: 0x40XX 00XX

The reset value of HSICAL[7:0] and MSICAL[7:0] is factory-programmed.

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	HSITRIM[6:0]							HSICAL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSITRIM[7:0]								MSICAL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSITRIM[6:0]:** HSI16 clock trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. They can be programmed to adjust to variations in voltage and temperature that influence the HSI16 frequency.

The default value is 64 that, when added to the HSICAL value, must trim the HSI16 to 16 MHz ± 1 %.

Bits 23:16 **HSICAL[7:0]:** HSI16 clock calibration

These bits are initialized at startup with the factory-programmed HSI16 calibration trim value. When HSITRIM is written, HSICAL is updated with the sum of HSITRIM and the factory trim value.

Bits 15:8 **MSITRIM[7:0]:** MSI clock trimming

These bits provide an additional user-programmable trimming value that is added to the MSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the MSI.

The default value is 0, that, when added to the MSICAL value, must trim the MSI to its mid frequency.

Bits 7:0 **MSICAL[7:0]:** MSI clock calibration

These bits are initialized at startup with the factory-programmed MSI calibration trim value. When MSITRIM is written, MSICAL is updated with the sum of MSITRIM and the factory trim value.

Note: Adding a MSITRIM value with the MSB set results in a subtraction.

6.4.3 RCC clock configuration register (RCC_CFGR)

Address offset: 0x008

Reset value: 0x0007 0000

(after POR reset and after wakeup from Standby)

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

One or two wait states inserted only if the access occurs during clock source switch.

From 0 to 15 wait states inserted if the access occurs when the APB or AHB prescalers values update is ongoing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			MCOSEL[3:0]				Res.	Res.	Res.	Res.	Res.	PPRE 2F	PPRE 1F	HPREF
	rw	rw	rw	rw	rw	rw	rw						r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP WUCK	Res.	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **MCOPRE[2:0]:** Microcontroller clock output prescaler

These bits are set and cleared by software.
It is highly recommended to change this prescaler before MCO output is enabled.
000: MCO divided by 1
001: MCO divided by 2
010: MCO divided by 4
011: MCO divided by 8
100: MCO divided by 16
Others: not allowed

Bits 27:24 **MCOSEL[3:0]:** Microcontroller clock output selection
These bits are set and cleared by software.
0000: MCO output disabled, no clock on MCO
0001: SYSCLK system clock selected
0010: MSI clock selected.
0011: HSI16 clock selected.
0100: HSE32 clock selected (after stabilization)
0101: Main PLLRCLK clock selected
0110: LSI clock selected
1000: LSE clock selected
1101: Main PLLPCLK clock selected
1110: Main PLLQCLK clock selected
Others: reserved

Note: This clock output may have some truncated cycles at startup, entering and wakeup from low-power modes, or during MCO clock source switching.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **PPRE2F:** PCLK2 prescaler flag (APB2)
This bit is set and reset by hardware to acknowledge PCLK2 prescaler programming. It is reset when a new prescaler value is programmed in PPRE2 and set when the programmed value is actually applied.
0: PCLK2 prescaler value not yet applied
1: PCLK2 prescaler value applied

Bit 17 **PPRE1F:** PCLK1 prescaler flag (APB1)
This bit is set and reset by hardware to acknowledge PCLK1 prescaler programming. It is reset when a new prescaler value is programmed in PPRE1 and set when the programmed value is actually applied.
0: PCLK1 prescaler value not yet applied
1: PCLK1 prescaler value applied

Bit 16 **HPREF:** HCLK1 prescaler flag (CPU, AHB1, and AHB2)
This bit is set and reset by hardware to acknowledge HCLK1 prescaler programming. It is reset when a new prescaler value is programmed in HPRE and set when the programmed value is actually applied.
0: HCLK1 prescaler value not yet applied
1: HCLK1 prescaler value applied

Bit 15 **STOPWUCK:** Wakeup from Stop and CSS backup clock selection
This bit is set and cleared by software to select the system clock used when exiting Stop mode. The selected clock is also used as emergency clock for the CSS on HSE32.
0: MSI oscillator selected as wakeup from stop clock and CSS backup clock.
1: HSI16 oscillator selected as wakeup from stop clock and CSS backup clock

Note: Warning: STOPWUCK must not be modified when the HSE32 CSS is enabled by CSSON in the RCC_CR register and the system clock is HSE32 (SWS = 10) or a switch on HSE32 is requested (SW = 10).

Bit 14 Reserved, must be kept at reset value.

Bits 13:11 **PPRE2[2:0]:** PCLK2 high-speed prescaler (APB2)

These bits are set and cleared by software to control the division factor of the PCLK2 clock (APB2). The PPRE2F flag can be checked to know if the programmed PPRE2 prescaler value is applied.

0xx: HCLK1 not divided
100: HCLK1 divided by 2
101: HCLK1 divided by 4
110: HCLK1 divided by 8
111: HCLK1 divided by 16

Bits 10:8 **PPRE1[2:0]:** PCLK1 low-speed prescaler (APB1)

These bits are set and cleared by software to control the division factor of the PCLK1 clock (APB1). The PPRE1F flag can be checked to know if the programmed PPRE1 prescaler value is applied.

0xx: HCLK1 not divided
100: HCLK1 divided by 2
101: HCLK1 divided by 4
110: HCLK1 divided by 8
111: HCLK1 divided by 16

Bits 7:4 **HPRE[3:0]:** HCLK1 prescaler (CPU, AHB1, and AHB2.)

These bits are set and cleared by software to control the division factor of the HCLK1 clock (CPU, AHB1, AHB2). The HPREF flag can be checked to know if the programmed HPRE prescaler value is applied.

0001: SYSCLK divided by 3
0010: SYSCLK divided by 5
0101: SYSCLK divided by 6
0110: SYSCLK divided by 10
0111: SYSCLK divided by 32
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512
Others: SYSCLK not divided

Caution: Depending on the device voltage range, the software must set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details please refer to *Section 5.1.4: Dynamic voltage scaling management*). After a write operation to these bits and before decreasing the voltage range, the HPREF bit must be read to be sure that the new value has been taken into account.

Bits 3:2 **SWS[1:0]:** System clock switch status

These bits are set and cleared by hardware to indicate which clock source is used as system clock.

00: MSI oscillator used as system clock
01: HSI16 oscillator used as system clock
10: HSE32 used as system clock
11: PLLRCLK used as system clock

Bits 1:0 **SW[1:0]:** System clock switch

These bits are set and cleared by software to select system clock source (SYSCLK). They are configured by hardware to force MSI oscillator selection when exiting Shutdown mode and Standby mode.

These bits can also be configured by hardware to force MSI or HSI16 oscillator selection when exiting Stop mode or in case of failure of the HSE32 oscillator, depending on STOPWUCK value.

00: MSI selected as system clock

01: HSI16 selected as system clock

10: HSE32 selected as system clock

11: PLLRCLK selected as system clock

6.4.4 RCC PLL configuration register (RCC_PLLCFGR)

Address offset: 0x00C

Reset value: 0x2204 0100

Access: no wait state, word, half-word and byte access

This register is used to configure the main PLL clock outputs according to the formulas:

- f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)
- f(PLL_P) = f(VCO clock) / PLLP
- f(PLL_Q) = f(VCO clock) / PLLQ
- f(PLL_R) = f(VCO clock) / PLLR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLR[2:0]			PLL REN	PLLQ[2:0]			PLL QEN	Res.	Res.	PLLP[4:0]					PLL PEN
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PLLN[6:0]							Res.	PLLM[2:0]			Res.	Res.	PLLSRC[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:29 **PLLR[2:0]:** Main PLL division factor for PLLRCLK

These bits are set and cleared by software to control the frequency of the main PLL output clock PLLRCLK. This output can be selected as system clock. These bits can be written only if PLL is disabled.

PLLRCLK output clock frequency = VCO frequency / PLLR with PLLR = 2, 3, 4,... or 8 [VCO frequency / (N + 1)]

000: reserved

001: PLLR = 2

010: PLLR = 3

011: PLLR = 4

100: PLLR = 5

101: PLLR = 6

110: PLLR = 7

111: PLLR = 8

Note: The software must set these bits correctly not to exceed 48 MHz on this domain in range 1.

Bit 28 **PLLREN:** Main PLL PLLRCLK output enable

This bit is set and reset by software to enable the PLLRCLK output of the main PLL. It cannot be written when PLLRCLK output of the PLL is used as system clock. In order to save power, when the PLLRCLK output of the PLL is not used, the value of PLLREN must be 0.

0: PLLRCLK output disabled

1: PLLRCLK output enabled

Bits 27:25 **PLLQ[2:0]:** Main PLL division factor for PLLQCLK

These bits are set and cleared by software to control the frequency of the main PLL output clock PLLQCLK. This output can be selected for True RNG clock. These bits can be written only if PLL is disabled.

PLLQCLK output clock frequency = VCO frequency / PLLQ with PLLQ = 2, 3, 4,... or 8 [VCO frequency / (N + 1)]

000: reserved

001: PLLQ = 2

010: PLLQ = 3

011: PLLQ = 4

100: PLLQ = 5

101: PLLQ = 6

110: PLLQ = 7

111: PLLQ = 8

Note: The software must set these bits correctly not to exceed 48 MHz on this domain in range 1.

Bit 24 **PLLQEN:** Main PLL PLLQCLK output enable

This bit is set and reset by software to enable the PLLQCLK output of the main PLL. In order to save power, when the PLLQCLK output of the PLL is not used, the value of PLLQEN must be 0.

0: PLLQCLK output disabled

1: PLLQCLK output enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:17 **PLLP[4:0]:** Main PLL division factor for PLLPCLK

These bits are set and cleared by software to control the frequency of the main PLL output clock PLLPCLK. This output can be selected for ADC. These bits can be written only if the PLL is disabled.

PLLPCLK output clock frequency = VCO frequency / PLLP with PLLP = 2, 3, 4, ...or 32 [VCO frequency / (N + 1)]

0000: reserved

00001: PLLP = 2

00010: PLLP = 3

00011: PLLP = 4

00100: PLLP = 5

...

11111: PLLP = 32

Caution: The software must set these bits correctly not to exceed 48 MHz on this domain in range 1.

Bit 16 **PLLPEN:** Main PLL PLLPCLK output enable

This bit is set and reset by software to enable the PLLPCLK output of the main PLL. In order to save power, when the PLLPCLK output of the PLL is not used, the value of PLLPEN must be 0.

0: PLLPCLK output disabled

1: PLLPCLK output enabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLN[6:0]:** Main PLL multiplication factor for VCO

These bits are set and cleared by software to control the multiplication factor of the VCO. They can be written only when the PLL is disabled.

VCO output frequency = VCO input frequency x PLLN with 6≤ PLLN ≤ 127

0000000: Reserved must not be used.

...

0000101: Reserved (must not be used)

0000110: PLLN = 6

0000111: PLLN = 7

...

1010101: PLLN = 85

1010110: PLLN = 86

...

1111111: PLLN = 127

Caution: The software must set correctly these bits to assure that the VCO output frequency is between 96 and 344 MHz.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PLLM[2:0]:** Division factor for the main PLL input clock

These bits are set and cleared by software to divide the PLL input clock before the VCO. They can be written only when the PLL is disabled.

VCO input frequency = PLL input clock frequency / PLLM with 1 ≤ PLLM ≤ 8

000: PLLM = 1

001: PLLM = 2

010: PLLM = 3

011: PLLM = 4

100: PLLM = 5

101: PLLM = 6

110: PLLM = 7

111: PLLM = 8

Caution: The software must set these bits correctly to ensure that the VCO input frequency ranges from 2.66 to 16 MHz.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSRC[1:0]:** Main PLL entry clock source

These bits are set and cleared by software to select PLL clock source. They can be written only when PLL is disabled. In order to save power, when no PLL is used, the value of PLLSRC must be 0.

00: No clock sent to PLL

01: MSI clock selected as PLL clock entry

10: HSI16 clock selected as PLL clock entry

11: HSE32 clock selected as PLL clock entry

6.4.5 RCC clock interrupt enable register (RCC_CIER)

Address offset: 0x018

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LSE CSSIE	Res.	Res.	Res.	PLL RDYIE	HSE RDYIE	HSI RDYIE	MSI RDYIE	LSE RDYIE	LSI RDYIE
						rw				rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **LSECSSIE:** LSE clock security system interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the CSS on LSE.

0: Clock security interrupt caused by LSE clock failure disabled

1: Clock security interrupt caused by LSE clock failure enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **PLLRDYIE:** PLL ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by PLL lock.

0: PLL lock interrupt disabled

1: PLL lock interrupt enabled

Bit 4 **HSERDYIE:** HSE32 ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the HSE32 oscillator stabilization.

0: HSE32 ready interrupt disabled

1: HSE32 ready interrupt enabled

Bit 3 **HSIRDYIE:** HSI16 ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the HSI16 oscillator stabilization.

0: HSI16 ready interrupt disabled

1: HSI16 ready interrupt enabled

Bit 2 **MSIRDYIE:** MSI ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the MSI oscillator stabilization.

0: MSI ready interrupt disabled

1: MSI ready interrupt enabled

Bit 1 **LSERDYIE:** LSE ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.

0: LSE ready interrupt disabled

1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE:** LSI ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the LSI oscillator stabilization.

0: LSI ready interrupt disabled

1: LSI ready interrupt enabled

6.4.6 RCC clock interrupt flag register (RCC_CIFR)

Address offset: 0x01C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LSE CSSF	CSSF	Res.	Res.	PLL RDYF	HSE RDYF	HSI RDYF	MSI RDYF	LSE RDYF	LSI RDYF
						r	r			r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **LSECSSF:** LSE CSS (clock security system) flag after masking

This bit is set by hardware when LSECSSIE = 1 and a failure is detected in the LSE oscillator. It is cleared by software setting the LSECSSC bit.

0: No CSS interrupt caused by LSE clock failure

1: CSS interrupt caused by LSE clock failure

Bit 8 **CSSF:** HSE32 CSS flag

This bit is set by hardware when a failure is detected in the HSE32 oscillator. It is cleared by software setting the CSSC bit.

0: No clock security interrupt caused by HSE32 clock failure

1: Clock security interrupt caused by HSE32 clock failure

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLLRDYF:** PLL ready interrupt flag

This bit is set by hardware when the PLL locks and PLLRDYDIE is set. It is cleared by software setting the PLLRDYC bit.

0: No clock ready interrupt caused by PLL lock

1: Clock ready interrupt caused by PLL lock

Bit 4 **HSERDYF:** HSE32 ready interrupt flag

This bit is set by hardware when the HSE32 clock becomes stable and HSERDYDIE is set. It is cleared by software setting the HSERDYC bit.

0: No clock ready interrupt caused by the HSE32 oscillator

1: Clock ready interrupt caused by the HSE32 oscillator

Bit 3 **HSIRDYF:** HSI16 ready interrupt flag

This bit is set by hardware when the HSI16 clock becomes stable and HSIRDYDIE is set in a response to setting the HSION in the RCC_CR register. When HSION is not set but the HSI16 oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated. This bit is cleared by software setting the HSIRDYC bit.

0: No clock ready interrupt caused by the HSI16 oscillator

1: Clock ready interrupt caused by the HSI16 oscillator

Bit 2 **MSIRDYF:** MSI ready interrupt flag

This bit is set by hardware when the MSI clock becomes stable and MSIRDYDIE is set. It is cleared by software setting the MSIRDYC bit.

0: No clock ready interrupt caused by the MSI oscillator

1: Clock ready interrupt caused by the MSI oscillator

Bit 1 **LSERDYF:** LSE ready interrupt flag

This bit is set by hardware when the LSE clock becomes stable and LSERDYDIE is set. It is cleared by software setting the LSERDYC bit.

0: No clock ready interrupt caused by the LSE oscillator

1: Clock ready interrupt caused by the LSE oscillator

Bit 0 **LSIRDYF:** LSI ready interrupt flag

This bit is set by hardware when the LSI clock becomes stable and LSIRDYDIE is set. It is cleared by software setting the LSIRDYC bit.

0: No clock ready interrupt caused by the LSI oscillator

1: Clock ready interrupt caused by the LSI oscillator

6.4.7 RCC clock interrupt clear register (RCC_CICR)

Address offset: 0x020

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LSE CSSC	CSSC	Res.	Res.	PLL RDYC	HSE RDYC	HSI RDYC	MSI RDYC	LSE RDYC	LSI RDYC
						w	w			w	w	w	w	w	w

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **LSECSSC:** LSE CSS flag clear
This bit is set by software to clear the LSECSSF flag.
0: No effect
1: LSECSSF flag cleared

Bit 8 **CSSC:** HSE32 CSS flag clear
This bit is set by software to clear the HSE32 CSSF flag.
0: No effect
1: HSE32 CSSF flag cleared

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLLRDYC:** PLL ready interrupt clear
This bit is set by software to clear the PLLRDYF flag.
0: No effect
1: PLLRDYF flag cleared

Bit 4 **HSERDYC:** HSE32 ready interrupt clear
This bit is set by software to clear the HSERDYF flag.
0: No effect
1: HSERDYF flag cleared

Bit 3 **HSIRDYC:** HSI16 ready interrupt clear
This bit is set software to clear the HSIRDYF flag.
0: No effect
1: HSIRDYF flag cleared

Bit 2 **MSIRDYC:** MSI ready interrupt clear
This bit is set by software to clear the MSIRDYF flag.
0: No effect
1: MSIRDYF flag cleared

Bit 1 **LSERDYC:** LSE ready interrupt clear
This bit is set by software to clear the LSERDYF flag.
0: No effect
1: LSERDYF flag cleared

Bit 0 **LSIRDYC:** LSI ready interrupt clear
This bit is set by software to clear the LSIRDYF flag.
0: No effect
1: LSIRDYF flag cleared

6.4.8 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x028

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRC RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMA MUX1 RST	DMA2 RST	DMA1 RST
			rw										rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST:** CRC reset
This bit is set and cleared by software.
0: No effect
1: CRC reset

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1RST:** DMAMUX1 reset
This bit is set and cleared by software.
0: No effect
1: DMAMUX1 reset

Bit 1 **DMA2RST:** DMA2 reset
This bit is set and cleared by software.
0: No effect
1: DMA2 reset

Bit 0 **DMA1RST:** DMA1 reset
This bit is set and cleared by software.
0: No effect
1: DMA1 reset

6.4.9 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x02C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOH RST	Res.	Res.	Res.	Res.	GPIOC RST	GPIOB RST	GPIOA RST
								rw					rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRST:** IO port H reset
This bit is set and cleared by software.
0: No effect
1: IO port H reset

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **GPIOCRST:** IO port C reset
This bit is set and cleared by software.
0: No effect
1: IO port C reset

Bit 1 **GPIOBRST:** IO port B reset
This bit is set and cleared by software.
0: No effect
1: IO port B reset

Bit 0 **GPIOARST:** IO port A reset
This bit is set and cleared by software.
0: No effect
1: IO port A reset

6.4.10 RCC AHB3 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x030

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLASH RST	Res.	Res.	Res.	Res.	Res.	HSEM RST	RNG RST	AES RST	PKA RST
						rw						rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHRST:** Flash interface reset
This bit can only be set when the Flash memory is in power down. It is set and cleared by software.
0: No effect
1: Flash memory interface reset

Bits 24:20 Reserved, must be kept at reset value.

Bit 19 **HSEMRST:** HSEM reset
This bit is set and cleared by software.
0: No effect
1: HSEM reset

Bit 18 **RNGRST:** True RNG reset
This bit is set and cleared by software.
0: No effect
1: True RNG reset

Bit 17 **AESRST:** AES hardware accelerator reset
This bit is set and cleared by software.
0: No effect
1: AES reset

Bit 16 **PKARST:** PKA hardware accelerator reset
This bit is set and cleared by software. PKA reset is disabled when a hardware PKA SRAM erase is ongoing.
0: No effect
1: PKA reset

Bits 15:0 Reserved, must be kept at reset value.

6.4.11 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 RST	Res.	DAC RST	Res.	Res.	Res.	Res.	Res.	I2C3 RST	I2C2 RST	I2C1 RST	Res.	Res.	Res.	USART2 RST	Res.
rw		rw						rw	rw	rw				rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2S2 RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2 RST
	rw														rw

Bit 31 **LPTIM1RST:** Low-power timer 1 reset
This bit is set and cleared by software.
0: No effect
1: LPTIM1 reset

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DACRST:** DAC reset
This bit is set and cleared by software.
0: No effect
1: DAC reset

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **I2C3RST:** I2C3 reset
This bit is set and cleared by software.
0: No effect
1: I2C3 reset

Bit 22 **I2C2RST:** I2C2 reset
This bit is set and cleared by software.
0: No effect
1: I2C2 reset

Bit 21 **I2C1RST:** I2C1 reset
This bit is set and cleared by software.
0: No effect
1: I2C1 reset

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **USART2RST:** USART2 reset
This bit is set and cleared by software.
0: No effect
1: USART2 reset

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **SPI2S2RST:** SPI2S2 reset
This bit is set and cleared by software.
0: No effect
1: SPI2S2 reset

Bits 13:1 Reserved, must be kept at reset value.

Bit 0 **TIM2RST:** TIM2 timer reset
This bit is set and cleared by software.
0: No effect
1: TIM2 reset

6.4.12 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x03C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM3RST	LPTIM2RST	Res.	Res.	Res.	Res.	LPUART1RST
									rw	rw					rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LPTIM3RST:** Low-power timer 3 reset
This bit is set and cleared by software.
0: No effect
1: LPTIM3 reset

Bit 5 **LPTIM2RST:** Low-power timer 2 reset
This bit is set and cleared by software.
0: No effect
1: LPTIM2 reset

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPUART1RST:** Low-power UART 1 reset
This bit is set and cleared by software.
0: No effect
1: LPUART1 reset

6.4.13 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x040

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17 RST	TIM16 RST	Res.
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 RST	Res.	SPI1 RST	TIM1 RST	Res.	ADC RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw		rw									

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **TIM17RST:** Timer 17 reset
This bit is set and cleared by software.
0: No effect
1: TIM17 reset

Bit 17 **TIM16RST:** Timer 16 reset
This bit is set and cleared by software.
0: No effect
1: TIM16 reset

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1RST:** USART1 reset
This bit is set and cleared by software.
0: No effect
1: USART1 reset

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST:** SPI1 reset
This bit is set and cleared by software.
0: No effect
1: SPI1 reset

Bit 11 **TIM1RST:** Timer 1 reset
This bit is set and cleared by software.
0: No effect
1: TIM1 reset

Bit 10 Reserved, must be kept at reset value.

Bit 9 **ADCRST:** ADC reset
This bit is set and cleared by software.
0: No effect
1: ADC reset

Bits 8:0 Reserved, must be kept at reset value.

6.4.14 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x044

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBGHZSPIRST
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SUBGHZSPIRST:** Sub-GHz radio SPI reset
This bit is set and cleared by software.
0: No effect
1: Sub-GHz radio SPI reset

6.4.15 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x048

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRC EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMA MUX1 EN	DMA2 EN	DMA1 EN
			rw										rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN:** CRC clock enable
This bit is set and cleared by software.
0: CRC clock disabled
1: CRC clock enabled

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1EN:** DMAMUX1 clock enable
This bit is set and cleared by software.
0: DMAMUX1 clock disabled
1: DMAMUX1 clock enabled

Bit 1 **DMA2EN:** DMA2 clock enable
This bit is set and cleared by software.
0: DMA2 clock disabled
1: DMA2 clock enabled

Bit 0 **DMA1EN:** DMA1 clock enable
This bit is set and cleared by software.
0: DMA1 clock disabled
1: DMA1 clock enabled

6.4.16 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x04C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHEN	Res.	Res.	Res.	Res.	GPIOCEN	GPIOBEN	GPIOAEN
								rw					rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHEN:** IO port H clock enable
This bit is set and cleared by software.
0: IO port H clock disabled
1: IO port H clock enabled

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **GPIOCEN:** IO port C clock enable
This bit is set and cleared by software.
0: IO port C clock disabled
1: IO port C clock enabled

Bit 1 **GPIOBEN:** IO port B clock enable
This bit is set and cleared by software.
0: IO port B clock disabled
1: IO port B clock enabled

Bit 0 **GPIOAEN:** IO port A clock enable
This bit is set and cleared by software.
0: IO port A clock disabled
1: IO port A clock enabled

6.4.17 RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)

Address offset: 0x050

Reset value: 0x0208 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	HSEMEN	RNGEN	AESEN	PKAEN
						rw						rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHEN:** Flash memory interface clock enable
This bit can only be cleared when the Flash memory is in power down. Set and cleared by software.
0: Flash interface clock disabled
1: Flash interface clock enabled

Bits 24:20 Reserved, must be kept at reset value.

Bit 19 **HSEMEN:** HSEM clock enable
This bit is set and cleared by software.
0: HSEM clock disabled
1: HSEM clock enabled

Bit 18 **RNGEN:** true RNG clocks enable
This bit is set and cleared by software.
0: True RNG bus and kernel clocks disabled
1: True RNG bus and kernel clocks enabled

Bit 17 **AESEN:** AES accelerator clock enable
This bit is set and cleared by software.
0: AES clock disabled
1: AES clock enabled

Bit 16 **PKAEN:** PKA accelerator clock enable
This bit is set and cleared by software.
PKA clock is enabled when a hardware PKA SRAM erase is ongoing.
0: PKA clock disabled
1: PKA clock enabled

Bits 15:0 Reserved, must be kept at reset value.

6.4.18 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address offset: 0x058

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 EN	Res.	DAC EN	Res.	Res.	Res.	Res.	Res.	I2C3 EN	I2C2 EN	I2C1 EN	Res.	Res.	Res.	USART2 EN	Res.
rw		rw						rw	rw	rw				rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2S2 EN	Res.	Res.	WWDG EN	RTCAPBEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2 EN
	rw			rs	rw										rw

Bit 31 **LPTIM1EN:** Low power timer 1 clocks enable
This bit is set and cleared by software.
0: LPTIM1 bus and kernel clocks disabled
1: LPTIM1 bus and kernel clocks enabled

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DACEN:** DAC clock enable
This bit is set and cleared by software.
0: DAC clock disabled
1: DAC clock enabled

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **I2C3EN:** I2C3 clocks enable
This bit is set and cleared by software.
0: I2C3 bus and kernel clocks disabled
1: I2C3 bus and kernel clocks enabled

Bit 22 **I2C2EN:** I2C2 clocks enable
This bit is set and cleared by software.
0: I2C2 bus and kernel clocks disabled
1: I2C2 bus and kernel clocks enabled

Bit 21 **I2C1EN:** I2C1 clocks enable
This bit is set and cleared by software.
0: I2C1 bus and kernel clocks disabled
1: I2C1 bus and kernel clocks enabled

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **USART2EN:** USART2 clock enable
This bit is set and cleared by software.
0: USART2 bus and kernel clocks disabled
1: USART2 bus and kernel clocks enabled

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **SPI2S2EN:** SPI2S2 clock enable
This bit is set and cleared by software.
0: SPI2S2 clock disabled
1: SPI2S2 clock enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN:** Window watchdog clock enable
This bit is set by software to enable the window watchdog clock. It is reset by hardware system reset. This bit is forced to 1 by hardware when the hardware WWDG_SW option is reset.
0: Window watchdog clock disabled
1: Window watchdog clock enabled

Bit 10 **RTCAPBEN:** RTC APB bus clock enable
This bit is set and cleared by software.
RTC kernel clock is controlled by RCC_BDCR register bit RTCEN bit.
0: RTC APB bus clock disabled
1: RTC APB bus clock enabled

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **TIM2EN:** timer 2 clock enable
This bit is set and cleared by software.
0: TIM2 clock disabled
1: TIM2 clock enabled

6.4.19 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x05C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM3EN	LPTIM2EN	Res.	Res.	Res.	Res.	LPUART1EN
									rw	rw					rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LPTIM3EN:** Low-power timer 3 clocks enable
This bit is set and cleared by software.
0: LPTIM3 bus and kernel clocks disabled
1: LPTIM3 bus and kernel clocks enable

Bit 5 **LPTIM2EN:** Low-power timer 2 clocks enable
Set and cleared by software.
0: LPTIM2 bus and kernel clocks disable
1: LPTIM2 bus and kernel clocks enable

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPUART1EN:** Low power UART 1 clocks enable
Set and cleared by software.
0: LPUART1 bus and kernel clocks disable
1: LPUART1 bus and kernel clocks enable

6.4.20 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x060

Reset value: 0x0000 0000

Access: word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17 EN	TIM16 EN	Res.
													rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 EN	Res.	SPI1 EN	TIM1 EN	Res.	ADC EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw		rw									

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN:** Timer 17 clock enable
This bit is set and cleared by software.
0: TIM17 clock disabled
1: TIM17 clock enabled

Bit 17 **TIM16EN:** Timer 16 clock enable
This bit is set and cleared by software.
0: TIM16 clock disabled
1: TIM16 clock enabled

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1EN:** USART1 clocks enable
This bit is set and cleared by software.
0: USART1 bus and kernel clocks disabled
1: USART1 bus and kernel clocks enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN:** SPI1 clock enable
This bit is set and cleared by software.
0: SPI1 clock disabled
1: SPI1 clock enabled

Bit 11 **TIM1EN:** TIM1 timer clock enable
This bit is set and cleared by software.
0: TIM1 timer clock disabled
1: TIM1P timer clock enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **ADCEN:** ADC clocks enable
This bit is set and cleared by software.
0: ADC bus and kernel clocks disabled
1: ADC bus and kernel clocks enabled

Bits 8:0 Reserved, must be kept at reset value.

6.4.21 RCC APB3 peripheral clock enable register (RCC_APB3ENR)

Address offset: 0x64

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBGHZSPIEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SUBGHZSPIEN:** sub-GHz radio SPI clock enable
This bit is set and cleared by software.
0: Sub-GHz radio SPI clock disable
1: Sub-GHz radio SPI clock enable

6.4.22 RCC AHB1 peripheral clock enable in Sleep mode register (RCC_AHB1SMENR)

Address offset: 0x068

Reset value: 0x0000 1007

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRC SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAMUX1SMEN	DMA2SMEN	DMA1SMEN
			rw										rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CRCSMEN:** CRC clock enable during Sleep mode.
This bit is set and cleared by software.
0: CRC clock disabled by the clock gating during Sleep and Stop modes
1: CRC clock enabled by the clock gating during Sleep mode, disabled during Stop mode.

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1SMEN:** DMAMUX1 clock enable during Sleep mode.
This bit is set and cleared by software.
0: DMAMUX1 clock disabled by the clock gating during Sleep and Stop modes
1: DMAMUX1 clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 1 **DMA2SMEN:** DMA2 clock enable during Sleep mode
This bit is set and cleared by software.
0: DMA2 clock disabled by the clock gating during Sleep and Stop modes
1: DMA2 clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 0 **DMA1SMEN:** DMA1 clock enable during Sleep mode.
This bit is set and cleared by software.
0: DMA1 clock disabled by the clock gating during Sleep and Stop modes
1: DMA1 clock enabled by the clock gating during Sleep mode, disabled during Stop mode.

6.4.23 RCC AHB2 peripheral clock enable in Sleep mode register (RCC_AHB2SMENR)

Address offset: 0x06C

Reset value: 0x0000 0087

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHSMEN	Res.	Res.	Res.	Res.	GPIOCSMEN	GPIOBSMEN	GPIOASMEN
								rw					rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHSMEN:** IO port H clock enable during Sleep mode.
This bit is set and cleared by software.
0: IO port H clock disabled by the clock gating during Sleep and Stop modes
1: IO port H clock enabled by the clock gating during Sleep mode, disabled during Stop mode.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **GPIOCSMEN:** IO port C clock enable during Sleep mode.
This bit is set and cleared by software.
0: IO port C clock disabled by the clock gating during Sleep and Stop modes
1: IO port C clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 1 **GPIOBSMEN:** IO port B clock enable during Sleep mode.
This bit is set and cleared by software.
0: IO port B clock disabled by the clock gating during Sleep and Stop modes
1: IO port B clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 0 **GPIOASMEN:** IO port A clock enable during Sleep mode.
This bit is set and cleared by software.
0: IO port A clock disabled by the clock gating during Sleep and Stop modes
1: IO port A clock enabled by the clock gating during Sleep mode, disabled during Stop mode.

6.4.24 RCC AHB3 peripheral clock enable in Sleep and Stop mode register (RCC_AHB3SMENR)

Address offset: 0x070

Reset value: 0x0387 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLASH SMEN	SRAM2 SMEN	SRAM1 SMEN	Res.	Res.	Res.	Res.	RNG SMEN	AES SMEN	PKA SMEN
						rw	rw	rw					rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHSMEN:** Flash memory interface clock enable during Sleep mode.
This bit is set and cleared by software.
0: Flash memory interface clock disabled by the clock gating during Sleep and Stop modes
1: Flash memory interface clock enabled by the clock gating during Sleep mode, disabled during Stop mode.

Bit 24 **SRAM2SMEN:** SRAM2 memory interface clock enable during Sleep mode
This bit is set and cleared by software.
0: SRAM2 clock disabled by the clock gating during Sleep and Stop modes
1: SRAM2 clock enabled by the clock gating during Sleep mode, disabled during Stop mode.

Bit 23 **SRAM1SMEN:** SRAM1 interface clock enable during Sleep mode.
This bit is set and cleared by software.
0: SRAM1 interface clock disabled by the clock gating during Sleep and Stop modes
1: SRAM1 interface clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bits 22:19 Reserved, must be kept at reset value.

Bit 18 **RNGSMEN:** True RNG clocks enable during Sleep and Stop modes
This bit is set and cleared by software.
0: True RNG bus clock disabled by the clock gating during Sleep and Stop modes
1: True RNG bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode.

Bit 17 **AESSMEN:** AES accelerator clock enable during Sleep mode.
This bit is set and cleared by software.
0: AES clock disabled by the clock gating during Sleep and Stop modes
1: AES clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 16 **PKASMEN:** PKA accelerator clock enable during Sleep mode.
This bit is set and cleared by software.
0: PKA clock disabled by the clock gating during Sleep and Stop modes
1: PKA clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bits 15:0 Reserved, must be kept at reset value.

6.4.25 RCC APB1 peripheral clock enable in Sleep mode register 1 (RCC_APB1SMENR1)

Address offset: 0x078

Reset value: 0xA0E2 4C01

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 SMEN	Res.	DAC SMEN	Res.	Res.	Res.	Res.	Res.	I2C3 SMEN	I2C2 SMEN	I2C1 SMEN	Res.	Res.	Res.	USART2 SMEN	Res.
rw		rw						rw	rw	rw				rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2S2 SMEN	Res.	Res.	WWDG SMEN	RTC APB SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2 SMEN
	rw			rw	rw										rw

Bit 31 **LPTIM1SMEN:** Low power timer 1 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: LPTIM1 bus clock disabled by the clock gating during Sleep and Stop modes
1: LPTIM1 bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DACSMEN:** DAC clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: DAC clock disabled by the clock gating during Sleep and Stop modes
1: DAC clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **I2C3SMEN:** I2C3 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: I2C3 bus clock disabled by the clock gating during Sleep and Stop modes
1: I2C3 bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 22 **I2C2SMEN:** I2C2 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: I2C2 bus clock disabled by the clock gating during Sleep and Stop modes
1: I2C2 bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 21 **I2C1SMEN:** I2C1 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: I2C1 bus clock disabled by the clock gating during Sleep and Stop modes
1: I2C1 bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **USART2SMEN:** USART2 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: USART2 bus clock disabled by the clock gating during Sleep and Stop modes
1: USART2 bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **SPI2S2SMEN:** SPI2S2 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: SPI2S2 clock disabled by the clock gating during Sleep and Stop modes
1: SPI2S2 clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGSMEN:** Window watchdog clocks enable during Sleep and Stop modes
This bit is set and cleared by software.It is forced to 1 by hardware when the hardware WWDG_SW option is reset.
0: Window watchdog clock disabled by the clock gating during Sleep and Stop modes
1: Window watchdog clocks enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 10 **RTCAPBSMEN:** RTC APB bus clock enable during Sleep and Stop modes
This bit is set and cleared by software. RTC kernel clock is controlled by the RTCEN bit in the RCC_BDCR register.
0: RTC APB bus clock disabled during by the clock gating during Sleep and Stop modes
1: RTC APB bus clock enabled during by the clock gating during Sleep mode, disabled during Stop mode

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **TIM2SMEN:** Timer 2 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: TIM2 clock disabled by the clock gating during Sleep and Stop modes
1: TIM2 clock enabled by the clock gating during Sleep mode, disabled during Stop mode

6.4.26 RCC APB1 peripheral clock enable in Sleep mode register 2 (RCC_APB1SMENR2)

Address offset: 0x07C

Reset value: 0x0000 0061

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM3SMEN	LPTIM2SMEN	Res.	Res.	Res.	Res.	LPUART1SMEN
									rw	rw					rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LPTIM3SMEN:** Low-power timer 3 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: LPTIM3 bus clock disabled by the clock gating during Sleep and Stop modes
1: LPTIM3 bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 5 **LPTIM2SMEN:** Low power timer 2 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: LPTIM2 bus clock disabled by the clock gating during Sleep and Stop modes
1: LPTIM2 bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPUART1SMEN:** Low-power UART 1 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: LPUART1 bus clock disabled by the clock gating during Sleep and Stop modes
1: LPUART1 bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

6.4.27 RCC APB2 peripheral clock enable in Sleep mode register (RCC_APB2SMENR)

Address offset: 0x080

Reset value: 0x0006 5A00

Access: word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17 SMEN	TIM16 SMEN	Res.
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 SMEN	Res.	SPI1 SMEN	TIM1 SMEN	Res.	ADC SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw		rw									

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN:** Timer 17 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: TIM17 clock disabled by the clock gating during Sleep and Stop modes
1: TIM17 clocks enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 17 **TIM16SMEN:** Timer 16 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: TIM16 clock disabled by the clock gating during Sleep and Stop modes
1: TIM16 clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN:** USART1 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: USART1 bus clock disabled by the clock gating during Sleep and Stop modes
1: USART1 bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SMEN:** SPI1 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: SPI1 clock disabled by the clock gating during Sleep and Stop modes
1: SPI1 clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 11 **TIM1SMEN:** Timer 1 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: TIM1 clocks disabled by the clock gating during Sleep and Stop modes
1: TIM1 clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bit 10 Reserved, must be kept at reset value.

Bit 9 **ADCSMEN:** ADC clocks enable during Sleep and Stop modes
This bit is set and cleared by software.
0: ADC bus clock disabled by the clock gating during Sleep and Stop modes
1: ADC bus clock enabled by the clock gating during Sleep mode, disabled during Stop mode

Bits 8:0 Reserved, must be kept at reset value.

6.4.28 RCC APB3 peripheral clock enable in Sleep mode register (RCC_APB3SMENR)

Address offset: 0x084

Reset value: 0x0000 0001

Access: word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBGHZSPISMEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SUBGHZSPISMEN:** Sub-GHz radio SPI clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: Sub-GHz radio SPI clock disabled by the clock gating during Sleep and Stop modes
1: Sub-GHz radio SPI clock enabled by the clock gating during Sleep mode, disabled during Stop mode

6.4.29 RCC peripherals independent clock configuration register (RCC_CCIPR)

Address offset: 0x088

Reset value: 0x0000 0000

Access: no wait states, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNGSEL[1:0]		ADCSEL[1:0]		Res.	Res.	Res.	Res.	LPTIM3SEL[1:0]		LPTIM2SEL[1:0]		LPTIM1SEL[1:0]		I2C3SEL[1:0]	
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C2SEL[1:0]		I2C1SEL[1:0]		LPUART1SEL [1:0]		SPI2S2SEL [1:0]		Res.	Res.	Res.	Res.	USART2SEL [1:0]		USART1SEL [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:30 **RNGSEL[1:0]:** RNG clock source selection
These bits are set and cleared by software to select the clock source used by the true RNG.
00: PLL “Q” clock (PLLQCLK) selected
01: LSI clock selected
10: LSE clock selected
11: MSI clock selected

Bits 29:28 **ADCSEL[1:0]:** ADC clock source selection
These bits are set and cleared by software to select the clock source used by the ADC interface.
00: No clock selected
01: HSI16 clock selected
10: PLL “P” clock (PLLPCLK) selected
11: System clock (SYSCLK) selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:22 **LPTIM3SEL[1:0]:** Low-power timer 3 clock source selection
These bits are set and cleared by software to select the LPTIM3 clock source.
00: PCLK selected
01: LSI clock selected
10: HSI16 clock selected
11: LSE clock selected

Bits 21:20 **LPTIM2SEL[1:0]:** Low-power timer 2 clock source selection
These bits are set and cleared by software to select the LPTIM2 clock source.
00: PCLK selected
01: LSI clock selected
10: HSI16 clock selected
11: LSE clock selected

Bits 19:18 **LPTIM1SEL[1:0]:** Low-power timer 1 clock source selection
These bits are set and cleared by software to select the LPTIM1 clock source.
00: PCLK selected
01: LSI clock selected
10: HSI16 clock selected
11: LSE clock selected

Bits 17:16 **I2C3SEL[1:0]:** I2C3 clock source selection
These bits are set and cleared by software to select the I2C3 clock source.
00: PCLK selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: Reserved

Bits 15:14 **I2C2SEL[1:0]:** I2C2 clock source selection
These bits are set and cleared by software to select the I2C2 clock source.
00: PCLK selected
01: System clock (SYSCLK) selected
10: HSI16 clock
11: Reserved

Bits 13:12 **I2C1SEL[1:0]:** I2C1 clock source selection
These bits are set and cleared by software to select the I2C1 clock source.
00: PCLK selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: Reserved

Bits 11:10 **LPUART1SEL[1:0]:** LPUART1 clock source selection
These bits are set and cleared by software to select the LPUART1 clock source.
00: PCLK selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: LSE clock selected

Bits 9:8 **SPI2S2SEL[1:0]:** SPI2S2 I2S clock source selection
This bit is set and cleared by software to select the SPI2S2 I2S clock source.
00: Reserved
01: PLL “Q” clock (PLLQCLK) selected
10: HSI16 clock selected
11: External input I2S_CKIN selected

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **USART2SEL[1:0]:** USART2 clock source selection
This bit is set and cleared by software to select the USART2 clock source.
00: PCLK selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: LSE clock selected

Bits 1:0 **USART1SEL[1:0]:** USART1 clock source selection
These bits are set and cleared by software to select the USART1 clock source.
00: PCLK selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: LSE clock selected

6.4.30 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x090

Reset value: 0x0000 0000

Reset by Backup domain reset, except LSCOSEL, LSCOEN and BDRST that are reset only by Backup domain power-on reset but not reset by wakeup from Standby and NRST pad.

Access: 0 ≤ wait state ≤ 3, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

Note: The bits of this register are outside of the V_{CORE} domain. As a result, after Reset, these bits are write-protected and the DBP bit in the PWR control register 1 (PWR_CR1) must be set before these bits can be modified. Refer to Section 5.1.2: Battery Backup domain for further information. These bits (except LSCOSEL, LSCOEN and BDRST) are only reset after a Backup domain reset (see Section 6.1.3: Backup domain reset). Any internal or external reset has no effect on these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
						rw	rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	LSESYSRDY	Res.	RTCSEL[1:0]		LSESYSEN	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw				r		rw	rw	rw	r	rw	rw	rw	rw	r	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **LSCOSEL:** Low-speed clock output selection
This bit is set and cleared by software.
0: LSI clock selected
1: LSE clock selected

Bit 24 **LSCOEN:** Low-speed clock output enable
This bit is set and cleared by software.
0: LSCO disabled
1: LSCO enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **BDRST:** Backup domain software reset
This bit is set and cleared by software.
0: Reset not activated
1: Entire Backup domain reset

Bit 15 **RTCEN:** RTC kernel clock enable
This bit is set and cleared by software. The RTC APB bus clock is controlled by the RTCAPBEN bit in the RCC_APB1ENR1 register and the RTCAPBSMEN bit in the RCC_APB1SMENR1register.
0: RTC kernel clock disabled
1: RTC kernel clock enabled

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **LSESYSRDY:** LSE system clock ready

This bit is set and cleared by hardware to indicate when the LSE system clock is ready after the LSESYSEN bit is set. This bit is only valid when LSEON, LSERDY and LSESYSEN are set.

0: LSE system clock not ready

1: LSE system clock ready

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]:** RTC clock source selection

These bits are set by software to select the clock source for the RTC. Once the RTC clock source is selected, it cannot be changed anymore unless the Backup domain is reset, or unless a failure is detected on LSE (LSECSSD is set). The BDRST bit can be used to reset them.

These bits cannot be written when LSE clock security is enabled in LSECSSON.

00: No clock

01: LSE oscillator clock selected

10: LSI oscillator clock selected

11: HSE32 oscillator clock divided by 32 selected

Bit 7 **LSESYSEN:** LSE system clock enable

This bit is set and cleared by software to enabled the LSE as system clock to the USARTx, LPUARTx, LPTIMx, TIMx, RNG, system LSCO, MCO, MSI PLL mode. The LSE system clock is only enabled when LSEON and LSERDY are both set.

0: LSE system clock disabled to USARTx, LPUARTx, LPTIMx, TIMx, RNG, system LSCO, MCO, MSI PLL mode

1: LSE system clock enabled to USARTx, LPUARTx, LPTIMx, TIMx, RNG, system LSCO, MCO, MSI PLL mode

Note: The LSE clock for the RTC is not impacted by this bit.

Bit 6 **LSECSSD:** CSS on LSE failure detection

This bit is set by hardware to indicate when a failure is detected by the CSS on the external 32 kHz oscillator (LSE). This bit is only reset by hardware on a BDRST and a POR reset.

0: No failure detected on LSE (32 kHz oscillator)

1: Failure detected on LSE (32 kHz oscillator)

Bit 5 **LSECSSON:** CSS on LSE enable

This bit is set by software to enable the CSS on LSE (32 kHz oscillator). LSECSSON must be enabled after the LSE oscillator is enabled (LSEON bit = 1) and ready (LSERDY flag set by hardware), and after the RTCSEL bit is selected. This bit automatically enables the LSI oscillator. Once enabled this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case the software **must** disable the LSECSSON bit.

0: CSS on LSE off

1: CSS on LSE on

Bits 4:3 **LSEDRV[1:0]:** LSE oscillator drive capability

These bits are set by software to modulate the LSE oscillator drive capability.

00: Xtal mode lower driving capability

01: Xtal mode medium-low driving capability

10: Xtal mode medium-high driving capability

11: Xtal mode higher driving capability

Note: The oscillator is in Xtal mode when it is not in bypass mode.

Bit 2 **LSEBYP:** LSE oscillator bypass

This bit is set and cleared by software to bypass the LSE oscillator. It can be written only when the external 32 kHz oscillator is disabled (LSEON = 0 and LSERDY = 0).

0: LSE oscillator not bypassed

1: LSE oscillator bypassed

Bit 1 **LSERDY:** LSE oscillator ready

This bit is set and cleared by hardware to indicate when the external 32 kHz oscillator is stable.

0: LSE oscillator not ready

1: LSE oscillator ready

Note: Once the LSEON bit is cleared, this bit goes low after six LSE clock cycles.

Bit 0 **LSEON:** LSE oscillator enable

This bit is set and cleared by software.

0: LSE oscillator off

1: LSE oscillator on

Note: The LSE clock is directly forwarded to the RTC. To enable the LSE clock to other system peripherals (USARTx, LPUARTx, LPTIMx, TIMx, RNG, system LSCO, MCO, MSI PLL mode), LSE must be enabled with the LSESYSEN bit.

6.4.31 RCC control/status register (RCC_CSR)

Address offset: 0x094

Reset value: 0x0C01 C600

(Reset by NRST pad, except reset flags by POR only, not reset by wakeup from Standby)

Access: 0 ≤ wait state ≤ 3, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	BOR RSTF	PIN RSTF	OBLRS TF	RFILA RSTF	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFRST	RF RSTF	Res.	Res.	MSISRANGE[3:0]				Res.	Res.	Res.	LSI PRE	Res.	Res.	LSI RDY	LSION
rw	r			rw	rw	rw	rw				rw			r	rw

Bit 31 **LPWRRSTF:** Low-power reset flag

This bit is set by hardware when a reset occurs due to illegal Stop, Standby or Shutdown mode entry. It is cleared by writing to the RMVF bit.

0: No illegal mode reset occurred

1: Illegal mode reset occurred

Bit 30 **WWDGRSTF:** Window watchdog reset flag

This bit is set by hardware when a window watchdog reset occurs. It is cleared by writing to the RMVF bit.

0: No window watchdog reset occurred

1: Window watchdog reset occurred

Bit 29 **IWDGRSTF**: Independent window watchdog reset flag

This bit is set by hardware when an independent watchdog reset domain occurs. It is cleared by writing to the RMVF bit.

0: No independent watchdog reset occurred

1: Independent watchdog reset occurred

Bit 28 **SFTRSTF:** Software reset flag

This bit is set by hardware when a software reset occurs. It is cleared by writing to the RMVF bit.

0: No software reset occurred

1: Software reset occurred

Bit 27 **BORRSTF:** BOR flag

This bit is set by hardware when a BOR occurs. It is cleared by writing to the RMVF bit.

0: No BOR occurred

1: BOR occurred

Bit 26 **PINRSTF:** Pin reset flag

This bit is set by hardware when a reset from the NRST pin occurs. It is cleared by writing to the RMVF bit.

0: No reset from NRST pin occurred

1: Reset from NRST pin occurred

Bit 25 **OBLRSTF:** Option byte loader reset flag

This bit is set by hardware when a reset from the option byte loading occurs. It is cleared by writing to the RMVF bit.

0: No reset from option byte loading occurred

1: Reset from option byte loading occurred

Bit 24 **RFILARSTF:** Sub-GHz radio illegal command flag

This bit is set by hardware when a illegal sub-GHz radio command is sent. It is cleared by writing to the RMVF bit.

0: No sub-GHz radio illegal command occurred

1: Sub-GHz radio illegal command occurred

Bit 23 **RMVF:** Remove reset flag

This bit is set by software to clear the reset flags LPWRRSTF, WWDGRSTF, IWDGRSTF, SFTRSTF, BORRSTF, PINRSTF, OBLRSTF and RFILARSTF.

0: No effect

1: Reset flags reset

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **RFRST:** Sub-GHz radio reset
This bit is set and cleared by software.
0: Sub-GHz radio software reset removed
1: Sub-GHz radio software reset active

Bit 14 **RFRSTF:** Sub-GHz radio in reset status flag
This bit is set and cleared by hardware.
0: Sub-GHz radio out of reset
1: Sub-GHz radio in reset

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:8 **MSISRANGE[3:0]**: MSI clock ranges
These bits are configured by software and can be written only when MSIRGSEL = 1 in the RCC_CR register. They are used to choose the MSI frequency range when exiting Standby mode and when MSIRGSEL = 0.
0100: Range 4 around 1 MHz
0101: Range 5 around 2 MHz
0110: Range 6 around 4 MHz (reset value)
0111: Range 7 around 8 MHz
Others: Not allowed (hardware write protection)

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **LSIPRE:** LSI frequency prescaler
This bit is set and cleared by software. It can be written at any time, however a new value is only taken into account after a switching off and turning on sequence on LSI by using the LSION bit.
0: LSI clock not divided (LSI)
1: LSI clock divided by 128 (LSI / 128)

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **LSIRDY:** LSI oscillator ready
This bit is set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, this bit goes low after three LSI clock cycles. This bit can be set even if LSION = 0 if the LSI is requested by the CSS on LSE, by the independent watchdog or by the RTC.
0: LSI oscillator not ready
1: LSI oscillator ready

Bit 0 **LSION:** LSI oscillator enable
This bit is set and cleared by software.
0: LSI oscillator off
1: LSI oscillator on

6.4.32 RCC extended clock recovery register (RCC_EXTCFGR)

Address offset: 0x108

Reset value: 0x0003 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHDHPREF
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHDHPRE[3:0]			
												rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SHDHPREF:** HCLK3 shared prescaler flag (AHB3, Flash, SRAM1, and SRAM2)

This bit is set and cleared by hardware to acknowledge the shared HCLK3 prescaler programming. It is reset when a new prescaler value is programmed in SHDHPRE[3:0]. This bit is set when the programmed value is actually applied.

0: HCLK3 prescaler value not yet applied

1: HCLK3 prescaler value applied

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **SHDHPRE[3:0]:** HCLK3 shared prescaler (AHB3, Flash, SRAM1, and SRAM2)

These bits are set and cleared by software to control the division factor of the shared HCLK3 clock. (AHB3, Flash, SRAM1 and SRAM2, APB3). The SHDHPREF flag can be checked to know if the programmed SHDHPRE prescaler value is applied.

0001: SYSCLK divided by 3
0010: SYSCLK divided by 5
0101: SYSCLK divided by 6
0110: SYSCLK divided by 10
0111: SYSCLK divided by 32
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512
Others: SYSCLK not divided

Caution: Depending on the device voltage range, the software must set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (refer to *Section 5.1.4: Dynamic voltage scaling management*). After a write operation to these bits and before decreasing the voltage range, the SHDHPRE bit must be read to be sure that the new value is taken into account.

6.4.33 RCC register map

Table 55. RCC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	PLLRDY	PLLON	Res.	Res.	HSEBYPPWR	HSEPRE	CSSON	Res.	HSERDY	HSEON	Res.	Res.	Res.	HSIKERDY	HSIASFS	HSIRDY	HSIKERON	HSION	MSIRANGE [3:0]				MSIRGSEL	MSIPLLEN	MSIRDY	MSION
	Reset value							0	0			0	0	0		0	0				0	0	0	0	0	0	1	1	0	0	0	0	1
0x004	RCC_ICSCR	Res.	HSITRIM[6:0]							HSICAL[7:0]								MSITRIM[7:0]								MSICAL[7:0]							
	Reset value		1	0	0	0	0	0	0	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x
0x008	RCC_CFGR	Res.	MCOPRE [2:0]			MCOSEL [3:0]				Res.	Res.	Res.	Res.	Res.	PPRE2F	PPRE1F	HPREF	STOPWUCK	Res.	PPRE2[2:0]			PPRE1 [2:0]		HPRE[3:0]				SWS [1:0]		SW [1:0]	
	Reset value		0	0	0	0	0	0	0						1	1	1	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RCC_ PLLCFGR	PLLR[2:0]			PLLREN	PLLQ[2:0]			PLLQEN	Res.	Res.	PLLP[4:0]					PLLPEN	Res.	PLLN[6:0]							Res.	PLLM[2:0]			Res.	Res.	PLLSRC [1:0]	
	Reset value	0	0	1	0	0	0	1	0			0	0	0	1	0	0		0	0	0	0	0	0	1		0	0	0			0	0
0x010-0x014	Reserved	Reserved																															
0x018	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSIE	Res.	Res.	Res.	PLLRDYIE	HSERDYIE	HSIRDYIE	MSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value																							0				0	0	0	0	0	0
0x01C	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSF	CSSF	Res.	Res.	PLLRDYF	HSERDYF	HSIRDYF	MSIRDYF	LSERDYF	LSIRDYF
	Reset value																							0	0			0	0	0	0	0	0
0x020	RCC_CICR	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	LSECSSC	CSSC	Res.	Res.	PLLRDYC	HSERDYC	HSIRDYC	MSIRDYC	LSERDYC	LSIRDYC
	Reset value																							0	0			0	0	0	0	0	0
0x024	Reserved	Reserved																															
0x028	RCC_ AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAMUX1RST	DMA2RST	DMA1RST
	Reset value																				0										0	0	0
0x02C	RCC_ AHB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHRST	Res.	Res.	Res.	Res.	GPIOCRST	GPIOBRST	GPIOARST
	Reset value																									0					0	0	0

Table 55. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	RCC_ AHB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	FLASHRST	Res.	Res.	Res.	Res.	Res.	HSEMRST	RNGRST	AESRST	PKARST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							0						0	0	0	0																
0x034	Reserved	Reserved																															
0x038	RCC_ APB1RSTR1	LPTIM1RST	Res.	DACRST	Res.	Res.	Res.	Res.	Res.	I2C3RST	I2C2RST	I2C1RST	Res.	Res.	Res.	USART2RST	Res.	Res.	SPI2S2RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2RST
	Reset value	0		0						0	0	0				0			0													0
0x03C	RCC_ APB1RSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM3RST	LPTIM2RST	Res.	Res.	Res.	Res.	LPUART1RST
	Reset value																										0	0					0
0x040	RCC_ APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17RST	TIM16RST	Res.	Res.	USART1RST	Res.	SPI1RST	TIM1RST	Res.	ADCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0			0		0	0		0									
0x044	RCC_ APB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBGHZSPIRST
	Reset value																																0
0x048	RCC_ AHB1ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAMUX1EN	DMA2EN	DMA1EN
	Reset value																				0										0	0	0
0x04C	RCC_ AHB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHEN	Res.	Res.	Res.	Res.	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value																									0					0	0	0
0x050	RCC_ AHB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	HSEMEN	RNGEN	AESEN	PKAEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							1						1	0	0	0																
0x054	Reserved	Reserved																															
0x058	RCC_ APB1ENR1	LPTIM1EN	Res.	DACEN	Res.	Res.	Res.	Res.	Res.	I2C3EN	I2C2EN	I2C1EN	Res.	Res.	Res.	USART2EN	Res.	Res.	SPI2S2EN	Res.	Res.	WWDGEN	RTCAPBEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2EN
	Reset value	0		0						0	0	0				0			0			0	0									0

Table 55. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x05C	RCC_ APB1ENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM3EN	LPTIM2EN	Res.	Res.	Res.	Res.	LPUART1EN
	Reset value																										0	0					0
0x060	RCC_ APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	Res.	Res.	USART1EN	Res.	SPI1EN	TIM1EN	Res.	ADCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0			0		0	0		0										
0x064	RCC_ APB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBGHZSPIEN
	Reset value																																0
0x068	RCC_ AHB1SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAMUX1SMEN	DMA2SMEN	DMA1SMEN
	Reset value																				1										1	1	1
0x06C	RCC_ AHB2SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHSMEN	Res.	Res.	Res.	Res.	GPIOCSMEN	GPIOBSMEN	GPIOASMEN
	Reset value																									1					1	1	1
0x070	RCC_ AHB3SMENR	Res.	Res.	Res.	Res.	Res.	Res.	FLASHSMEN	SRAM2SMEN	SRAM1SMEN	Res.	Res.	Res.	Res.	RNGSMEN	AESSMEN	PKASMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							1	1	1					1	1	1																
0x074	Reserved	Reserved																															
0x078	RCC_APB1 SMENR1	LPTIM1SMEN	Res.	DACSMEN	Res.	Res.	Res.	Res.	Res.	I2C3SMEN	I2C2SMEN	I2C1SMEN	Res.	Res.	Res.	USART2SMEN	Res.	Res.	SPI2S2SMEN	Res.	Res.	WWDGSMEN	RTCAPBSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2SMEN
	Reset value	1		1						1	1	1				1			1			1	1										1
0x07C	RCC_APB1 SMENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM3SMEN	LPTIM2SMEN	Res.	Res.	Res.	Res.	LPUART1SMEN
	Reset value																										1	1					1

Table 55. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	RCC_APB2SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17SMEN	TIM16SMEN	Res.	Res.	USART1SMEN	Res.	SPI1SMEN	TIM1SMEN	Res.	ADCSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														1	1			1		1	1		1									
0x084	RCC_APB3SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBGHZSPISMEN
	Reset value																																1
0x088	RCC_CCIPR	RNGSEL [1:0]		ADCSEL [1:0]		Res.	Res.	Res.	Res.	LPTIM3SEL[1:0]		LPTIM2SEL [1:0]		LPTIM1SEL [1:0]		I2C3SEL [1:0]		I2C2SEL [1:0]		I2C1SEL [1:0]		LPUART1SEL [1:0]		SPI2S2SEL [1:0]		Res.	Res.	Res.	Res.	USART2SEL [1:0]		USART1SEL [1:0]	
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x08C	Reserved	Reserved																															
0x090	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	Res.	LSESYSRDY	Res.	RTCSEL[1:0]		LSESYSEN	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
	Reset value							0	0								0	0				0		0	0	0	0	0	0		0	0	0
0x094	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	RFILARSTF	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFRST	RFRSTF	Res.	Res.	MSISRANGE[3:0]				Res.	Res.	Res.	LSIPRE	Res.	Res.	LSIRDY	LSION
	Reset value	0	0	0	0	1	1	0	0	0								0	1			0	1	1	0				0			0	0
0x098-0x104	Reserved	Reserved																															
0x108	RCC_EXTCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHDHPREF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHDHPRE [3:0]			
	Reset value																1													0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

7 Hardware semaphore (HSEM)

7.1 Introduction

The hardware semaphore block provides 16 (32-bit) register based semaphores.

The semaphores can be used to ensure synchronization between different processes running on the core. The HSEM provides a non-blocking mechanism to lock semaphores in an atomic way. The following functions are provided:

- Semaphore lock, in two ways:
  - 2-step lock: by writing MASTERID and PROCID to the semaphore, followed by a read check
  - 1-step lock: by reading the MASTERID from the semaphore
- Interrupt generation when a semaphore is unlocked
  - Each semaphore may generate an interrupt
- Semaphore clear protection
  - A semaphore is only unlocked when MASTERID and PROCID match
- Global semaphore clear per MASTERID

7.2 Main features

The HSEM includes the following features:

- 16 (32-bit) semaphores
- 8-bit PROCID
- 4-bit MASTERID
- 1 interrupt line
- Lock indication

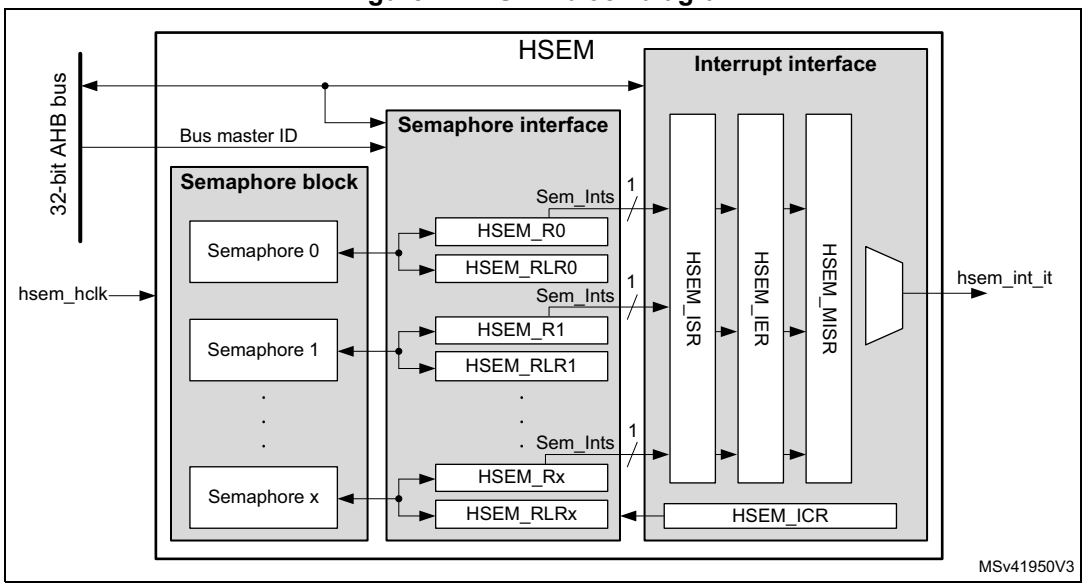
7.3 Functional description

7.3.1 HSEM block diagram

As shown in *Figure 27*, the HSEM is based on three sub-blocks:

- the semaphore block containing the semaphore status and IDs
- the semaphore interface block providing AHB access to the semaphore via the HSEM_Rx and HSEM_RLRx registers
- the interrupt interface block providing control for the interrupts via HSEM_ISR, HSEM_IER, HSEM_MISR, and HSEM_ICR registers.

Figure 27. HSEM block diagram



7.3.2 HSEM internal signals

Table 56. HSEM internal input/output signals

Signal name	Signal type	Description
AHB bus	Digital input/output	AHB register access bus
BusMasterID	Digital input	AHB bus master ID
hsem_int_it	Digital output	Interrupt line

7.3.3 HSEM lock procedures

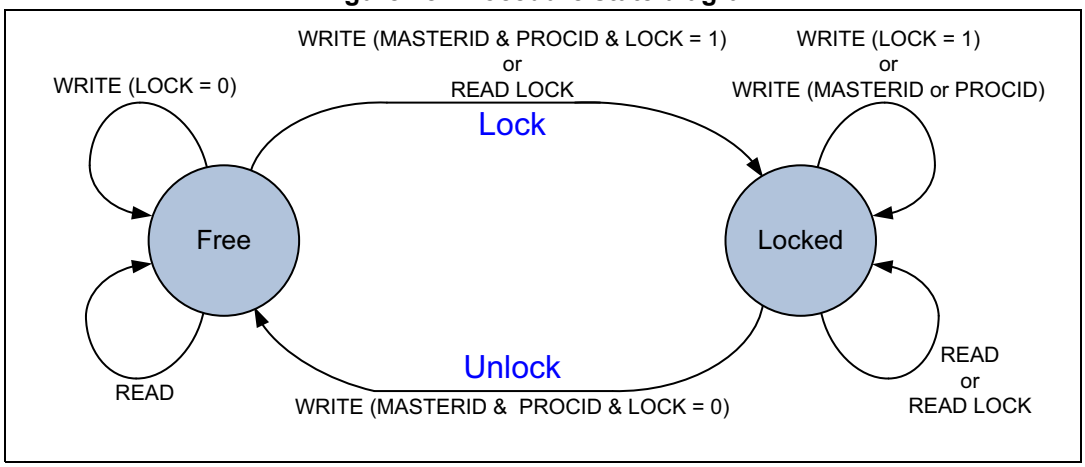
There are two lock procedures, namely 2-step (write) lock and 1-step (read) lock. The two procedures can be used concurrently.

The semaphore is free when its LOCK bit is 0. In this case, the MASTERID and PROCID are also 0. When the LOCK bit is 1, the semaphore is locked and the MASTERID indicates which AHB bus master ID has locked it. The PROCID indicates which process of that AHB bus master ID has locked the semaphore.

When write locking a semaphore, the written MASTERID must match the AHB bus master ID , and the PROCID is written by the AHB bus master software process taking the lock.

When read locking the semaphore, the MASTERID is taken from the AHB bus master ID, and the PROCID is forced to 0 by hardware. There is no PROCID available with read lock.

Figure 28. Procedure state diagram



2-step (write) lock procedure

The 2-step lock procedure consists in a write to lock the semaphore, followed by a read to check if the lock has been successful, carried out from the HSEM_Rx register

- Write semaphore with PROCID and MASTERID, and LOCK = 1. The MASTERID data written by software must match the AHB bus master information. i.e. a AHB bus master ID = 1writes data MASTERID = 1.

  Lock is put in place when the semaphore is free at write time.
- Read-back the semaphore

  The software checks the lock status, if PROCID and MASTERID match the written data, then the lock is confirmed.
- Else retry (the semaphore has been locked by another process).

A semaphore can only be locked when it is free.

A semaphore can be locked when the PROCID = 0.

Consecutive write attempts with LOCK = 1 to a locked semaphore are ignored.

1-step (read) lock procedure

The 1-step procedure consists in a read to lock and check the semaphore in a single step, carried out from the HSEM_RLRx register.

- Read lock semaphore with the AHB bus master MASTERID.
- If read MASTERID matches and PROCID = 0, then lock is put in place. If MASTERID matches and PROCID is not 0, this means that another process from the same MASTERID has locked the semaphore with a 2-step (write) procedure.
- Else retry (the semaphore has been locked by another process).

A semaphore can only be locked when it is free. When read locking a free semaphore, PROCID is 0. Read locking a locked semaphore returns the MASTERID and PROCID that locked it. All read locks, including the first one that locks the semaphore, return the MASTERID that locks or locked the semaphore.

Note: The 1-step procedure must not be used when running multiple processes of the same AHB bus master ID. All processes using the same semaphore read the same status. When only one process locks the semaphore, each process of that AHB bus master ID reads the semaphore as locked by itself with the MASTERID.

7.3.4 HSEM write/read/read lock register address

For each semaphore, two AHB register addresses are provided, separated in two banks of 32-bit semaphore registers, spaced by a 0x80 address offset.

In the first register address bank the semaphore can be written (locked/unlocked) and read through the HSEM_Rx registers.

In the second register address bank the semaphore can be read (locked) through the HSEM_RLRx registers.

7.3.5 HSEM unlock procedures

Unlocking a semaphore is a protected process, to prevent accidental clearing by a AHB bus master ID or by a process not having the semaphore lock right. The procedure consists in writing to the semaphore HSEM_Rx register with the corresponding MASTERID and PROCID and LOCK = 0. When unlocked the semaphore, the MASTERID, and the PROCID are all 0.

When unlocked, an interrupt may be generated to signal the event. To this end, the semaphore interrupt must be enabled.

The unlock procedure consists in a write to the semaphore HSEM_Rx register with matching MASTERID regardless on how the semaphore has been locked (1- or 2-step).

- Write semaphore with PROCID, MASTERID, and LOCK = 0
- If the written data matches the semaphore PROCID and MASTERID and the AHB bus master ID , the semaphore is unlocked and an interrupt may be generated when enabled, else write is ignored, semaphore remains locked and no interrupt is generated (the semaphore is locked by another process or the written data does not match the AHB bus master signaling).

Note: Different processes of the AHB bus master ID can write any PROCID value. Preventing other processes of the AHB bus master ID from unlocking a semaphore must be ensured by software, handling the PROCID correctly.

7.3.6 HSEM MASTERID semaphore clear

All semaphores locked by a MASTERID can be unlocked at once by using the HSEM_CR register. Write MASTERID and correct KEY value in HSEM_CR. All locked semaphores with a matching MASTERID are unlocked, and may generate an interrupt when enabled.

An interrupt may be generated for the unlocked semaphore(s). To this end, the semaphore interrupt must be enabled in the HSEM_IER register.

7.3.7 HSEM interrupts

An interrupt line hsem_int_it allows each semaphore to generate an interrupt.

An interrupt line provides the following features per semaphore:

- interrupt enable
- interrupt clear
- interrupt status
- masked interrupt status

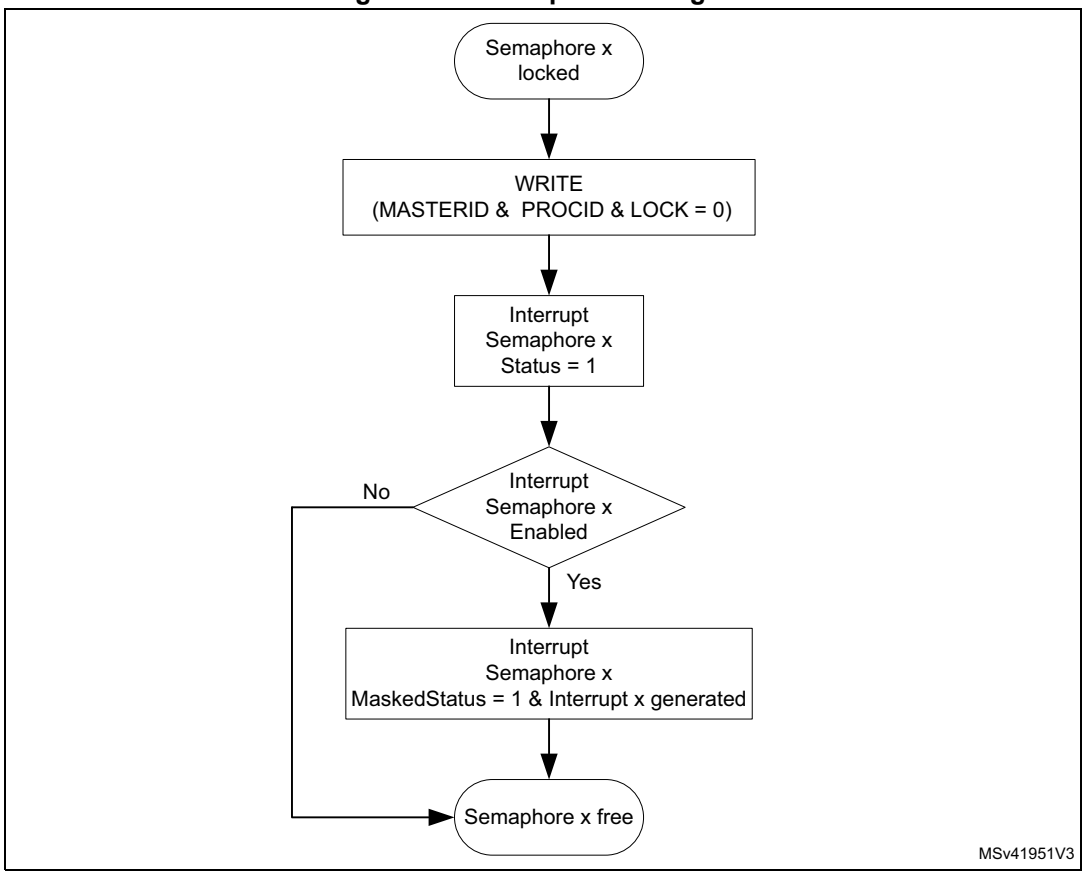
With the interrupt enable (HSEM_IER) the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts do not set the masked interrupt status MISF for that semaphore, and do not generate an interrupt on the interrupt line.

The interrupt clear (HSEM_ICR) clears the interrupt status ISF and masked interrupt status MISF of the associated semaphore for the interrupt line.

The interrupt status (HSEM_ISR) mirrors the semaphore interrupt status ISF before the enable.

The masked interrupt status (HSEM_MISR) only mirrors the semaphore enabled interrupt status MISF on the interrupt line. All masked interrupt status MISF of the enabled semaphores need to be cleared to clear the interrupt line.

Figure 29. Interrupt state diagram



The procedure to get an interrupt when a semaphore becomes free is described hereafter.

Try to lock semaphore x

- If the semaphore lock is obtained, no interrupt is needed.
- If the semaphore lock fails:
- Clear pending semaphore x interrupt status for the interrupt line in HSEM_ICR.
  Re-try to lock the semaphore x again:
  - If the semaphore lock is obtained, no interrupt is needed (semaphore has been freed between first try to lock and clear semaphore interrupt status).
  - If the semaphore lock fails, enable the semaphore x interrupt in HSEM_IER.

On semaphore x free interrupt, try to lock semaphore x

- If the semaphore lock is obtained:
  Disable the semaphore x interrupt in HSEM_IER.
  Clear pending semaphore x interrupt status in HSEM_ICR.
- If the semaphore x lock fails:
  Clear pending semaphore x interrupt status in HSEM_ICR.
  Try again to lock the semaphore x:
  - If the semaphore lock is obtained (semaphore has been freed between first try to lock and semaphore Interrupt status clear), disable the semaphore interrupt in HSEM_IER.

– If the semaphore lock fails, wait for semaphore free interrupt.

Note: An interrupt does not lock the semaphore. After an interrupt, either the AHB bus master or the process must still perform the lock procedure to lock the semaphore.

7.3.8 AHB bus master ID verification

The HSEM allows only authorized AHB bus master ID to lock and unlock semaphores.

- The AHB bus master 2-step lock write access to the semaphore HSEM_Rx register is checked against the valid bus master ID.
  – Accesses from unauthorized AHB bus master IDs are discarded and do not lock the semaphore.
- The AHB bus master 1-step lock read access from the semaphore HSEM_RLRx register is checked against the valid bus master ID.
  – An unauthorized AHB bus master ID read from HSEM_RLRx returns all 0.
- The semaphore unlock write access to the HSEM_CR register is checked against the valid bus master ID. Only the valid bus master ID can write to the HSEM_CR register and unlock any of the MASTERID semaphores.
  – Accesses from unauthorized AHB bus master IDs are discarded and do not clear the MASTERID semaphores.

Table 57 details the relation between bus master/processor and MASTERID.

Table 57. Authorized AHB bus master ID

Bus master 0 (CPU)
MASTERID = 4

Note: Accesses from unauthorized AHB bus master IDs to other registers are granted.

7.4 HSEM registers

Registers must be accessed using word format. Byte and half-word accesses are ignored and have no effect on the semaphores, they generate a bus error.

7.4.1 HSEM register semaphore x (HSEM_Rx)

Address offset: 0x000 + 0x4 * x (x = 0 to 15)

Reset value: 0x0000 0000

The HSEM_Rx must be used to perform a 2-step write lock, read back, and for unlocking a semaphore. Only write accesses with authorized AHB bus master ID is granted. Write accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID[7:0]							
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock indication
This bit can be written and read by software.
0: On write free semaphore (only when MASTERID and PROCID match), on read semaphore is free.
1: On write try to lock semaphore, on read semaphore is locked.

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **MASTERID[3:0]:** Semaphore MASTERID
Written by software
- When the semaphore is free and the LOCK bit is at the same time written to 1 and the MASTERID matches the AHB bus master ID.
- When the semaphore is unlocked (LOCK written to 0 and AHB bus master ID matched MASTERID, the MASTERID is cleared to 0.
- When the semaphore is unlocked (LOCK bit written to 0 or AHB bus master ID does not match MASTERID, the MASTERID is not affected.
- Write when LOCK bit is already 1 (semaphore locked), the MASTERID is not affected.
- An authorized read returns the stored MASTERID value.

Bits 7:0 **PROCID[7:0]:** Semaphore PROCID
Written by software
-When the semaphore is free and the LOCK is written to 1, and the MASTERID matches the AHB bus master ID, PROCID is set to the written data.
- When the semaphore is unlocked, LOCK written to 0 and AHB bus master ID matched MASTERID, the PROCID is cleared to 0.
- When the semaphore is unlocked, LOCK bit written to 0 and AHB bus master ID does not match MASTERID, the PROCID is not affected.
- Write when LOCK bit is already 1 (semaphore locked), the PROCID is not affected.
- An authorized read returns the stored PROCID value.

7.4.2 HSEM read lock register semaphore x (HSEM_RLRx)

Address offset: 0x080 + 0x4 * x (x = 0 to 15)

Reset value: 0x0000 0000

Accesses the same physical bits as HSEM_Rx. The HSEM_RLRx must be used to perform a 1-step read lock. Only read accesses with authorized AHB bus master ID is granted. Read accesses with unauthorized AHB bus master IDs are discarded and return 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **LOCK:** Lock indication

This bit is read only by software at this address.

- When the semaphore is free:

A read with a valid AHB bus master ID locks the semaphore and returns 1.

- When the semaphore is locked:

A read with a valid AHB bus master ID returns 1 (the MASTERID and PROCID reflect the already locked semaphore information).

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **MASTERID[3:0]:** Semaphore MASTERID

This field is read only by software at this address.

On a read, when the semaphore is free, the hardware sets the MASTERID to the AHB bus master ID reading the semaphore. The MASTERID of the AHB bus master locking the semaphore is read.

On a read when the semaphore is locked, this field returns the MASTERID of the AHB bus master that has locked the semaphore.

Bits 7:0 **PROCID[7:0]:** Semaphore processor ID

This field is read only by software at this address.

- On a read when the semaphore is free:

A read with a valid AHB bus master ID locks the semaphore and hardware sets the PROCID to 0.

- When the semaphore is locked:

A read with a valid AHB bus master ID returns the PROCID of the AHB bus master that has locked the semaphore.

7.4.3 HSEM interrupt enable register (HSEM_IER)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ISE[15:0]:** Interrupt semaphore x enable bit (x = 0 to 15)
This bit is read and written by software.
0: Interrupt generation for semaphore x disabled (masked)
1: Interrupt generation for semaphore x enabled (not masked)

7.4.4 HSEM interrupt clear register (HSEM_ICR)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISC[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ISC[15:0]:** Interrupt semaphore x clear bit (x = 0 to 15)
This bit is written by software, and is always read 0.
0: Interrupt semaphore x status ISFx and masked status MISFx not affected.
1: Interrupt semaphore x status ISFx and masked status MISFx cleared.

7.4.5 HSEM interrupt status register (HSEM_ISR)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ISF[15:0]:** Interrupt semaphore x status bit before enable (mask) (x = 0 to 15)
This bit is set by hardware, and reset only by software. This bit is cleared by software writing the corresponding HSEM_ICR bit.
0: Interrupt semaphore x status, no interrupt pending
1: Interrupt semaphore x status, interrupt pending

7.4.6 HSEM interrupt status register (HSEM_MISR)

Address offset: 0x10C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISF[15:0]:** Masked interrupt semaphore x status bit after enable (mask) (x = 0 to 15)
This bit is set by hardware and read only by software. This bit is cleared by software writing the corresponding HSEM_ICR bit. This bit is read as 0 when semaphore x status is masked in HSEM_IER bit x.
0: interrupt semaphore x status after masking not pending
1: interrupt semaphore x status after masking pending

7.4.7 HSEM clear register (HSEM_CR)

Address offset: 0x140

Reset value: 0x0000 0000

Only write accesses with authorized AHB bus master ID are granted. Write accesses with unauthorized AHB bus master ID are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MASTERID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				w	w	w	w								

Bits 31:16 **KEY[15:0]:** Semaphore clear key
This field can be written by software and is always read 0.
If this key value does not match HSEM_KEYR.KEY, semaphores are not affected.
If this key value matches HSEM_KEYR.KEY, all semaphores matching the MASTERID are cleared to the free state.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **MASTERID[3:0]:** MASTERID of semaphores to be cleared
This field can be written by software and is always read 0.
This field indicates the MASTERID for which the semaphores are cleared when writing the HSEM_CR.

Bits 7:0 Reserved, must be kept at reset value.

7.4.8 HSEM interrupt clear register (HSEM_KEYR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **KEY[15:0]:** Semaphore clear key
This field can be written and read by software.
Key value to match when clearing semaphores.

Bits 15:0 Reserved, must be kept at reset value.

7.4.9 HSEM register map

Table 58. HSEM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**HSEM_R0**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x004	**HSEM_R1**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
⋮																																	
0x03C	**HSEM_R15**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x080	**HSEM_RLR0**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID [3:0]				PROCID							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x084	**HSEM_RLR1**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
⋮																																	
0x0BC	**HSEM_RLR15**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x100	**HSEM_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISE[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	**HSEM_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	**HSEM_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C	**HSEM_MISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	**HSEM_CR**	KEY[15:0]																Res.	Res.	Res.	Res.	MASTERID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0								
0x144	**HSEM_KEYR**	KEY[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

Refer to *Section 2.4 on page 62* for the register boundary addresses.

8 General-purpose I/Os (GPIO)

8.1 GPIO introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR) and one 32-bit set/reset register (GPIOx_BSRR).

All GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

8.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allowing the use of I/O pins as GPIOs or as one of several peripheral functions

8.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of GPIO ports can be individually configured by software in several modes listed below:

- Input floating
- Input pull-up
- Input pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers must be accessed as 32-bit words, half-words or bytes.

GPIOx_BSRR and GPIOx_BRR registers allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 30 and *Figure 31* show the basic structure of a standard and a 5V-tolerant I/O port bit.

Table 59 gives the possible port bit configurations.

Figure 30. Basic structure of a standard I/O port bit

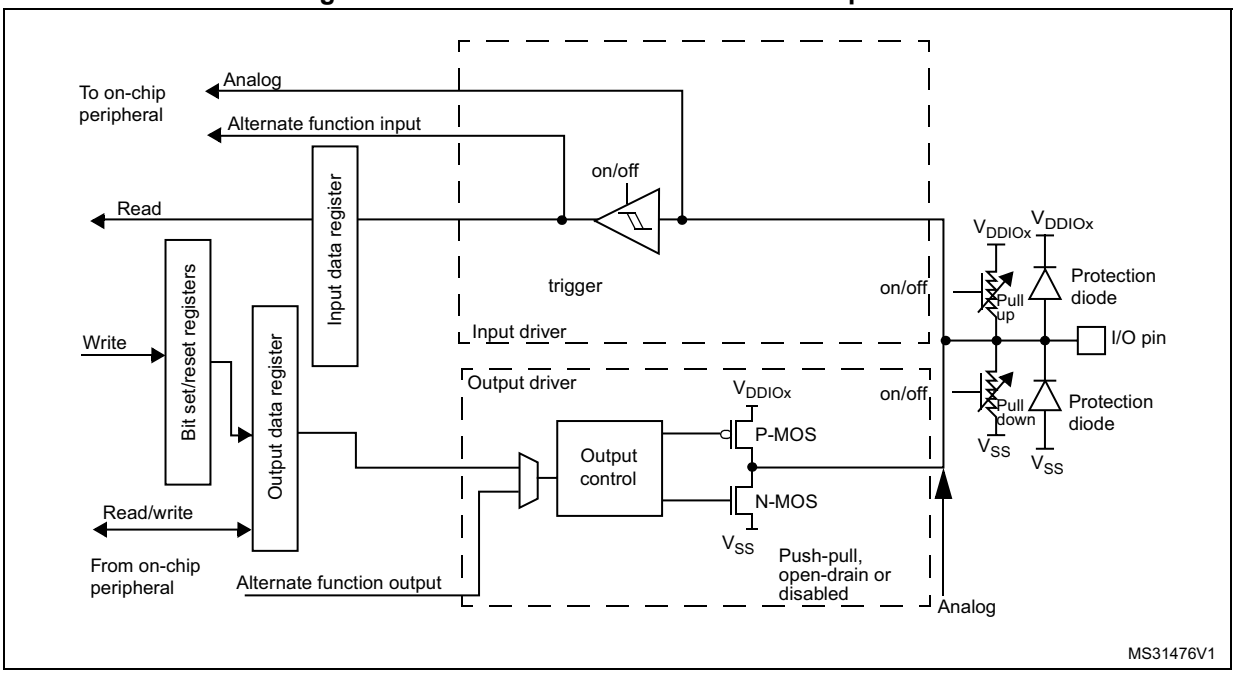
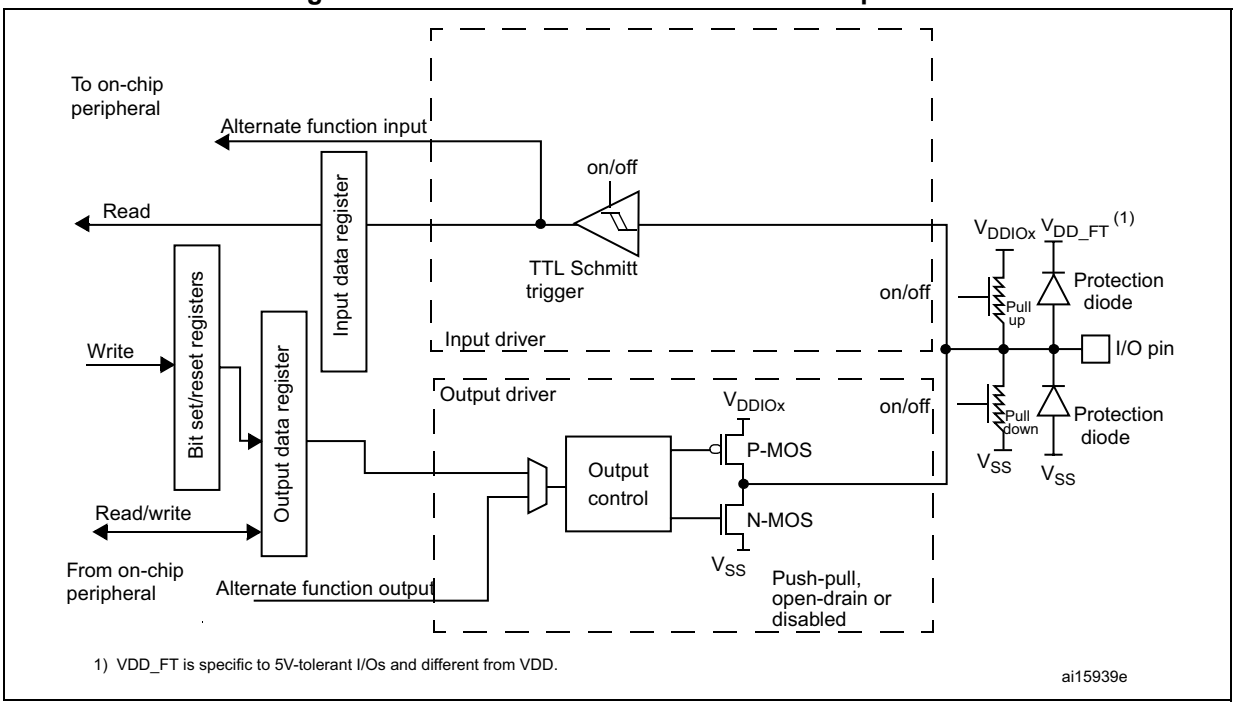


Figure 31. Basic structure of a 5V-tolerant I/O port bit



1) VDD_FT is specific to 5V-tolerant I/Os and different from VDD.

Table 59. Port bit configurations

MODE(i)[1:0]	OTYPER(i)	OSPEED(i)[1:0]	PUPD(i)[1:0]	I/O configuration(1)	
01	0	SPEED[1:0]	00	GP output	PP
	0		01	GP output	PP + PU
	0		10	GP output	PP + PD
	0		11	Reserved	
	1		00	GP output	OD
	1		01	GP output	OD + PU
	1		10	GP output	OD + PD
	1		11	Reserved (GP output OD)	
10	0		00	AF	PP
	0		01	AF	PP + PU
	0		10	AF	PP + PD
	0		11	Reserved	
	1		00	AF	OD
	1		01	AF	OD + PU
	1		10	AF	OD + PD
	1		11	Reserved	

Table 59. Port bit configurations (continued)

MODE(i)[1:0]	OTYPER(i)	OSPEED(i)[1:0]	PUPD(i)[1:0]	I/O configuration(1)	
00	X	XX	00	Input	Floating
	X	XX	01	Input	PU
	X	XX	10	Input	PD
	X	XX	11	Reserved (input floating)	
11	X	XX	00	Input/output	Analog
	X	XX	01	Reserved	
	X	XX	10		
	X	XX	11		

1. GP = general purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open drain, AF = alternate function.

8.3.1 General purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins listed below are in AF pull-up/pull-down after reset:

- PA15: JTDI in input mode with pull-up
- PA14: JTCK/SWCLK in input mode with pull-down
- PA13: JTMS/SWDAT in input mode with pull-up
- PB4: NJTRST in input mode with pull-up
- PB3: JTDO in HI-Z mode no pulls.

PH3/BOOT0 is in input mode during the reset until at least the end of the option byte loading phase (see *Section 8.3.15: Using PH3 as GPIO*).

When the I/O pin is configured as output, the value written to the output data register GPIOx_ODR is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register GPIOx_IDR captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

8.3.2 I/O pin alternate function multiplexer and mapping

The I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one AF peripheral connected to each I/O pin at a time. This avoids conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to sixteen alternate function inputs (AF0 to AF15), that can be configured through the registers GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15):

After reset, the multiplexer selection is alternate function 0 (AF0). I/Os are configured in alternate function mode through GPIOx_MODER register.

Specific alternate function assignments for each pin are detailed in the product datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped on different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user must proceed as follows:

- **Debug function**

  After each device reset, these pins are assigned as alternate function pins immediately usable by the debugger host.
- **GPIO**

  Configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function**
  - Connect the I/O to the desired AFx in the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDR registers, respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions**
  - For ADC, DAC and COMP, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC, DAC and COMP registers.
  - For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the product datasheet for the detailed mapping of the alternate function I/O pins.

8.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR) to configure up to sixteen I/Os.

GPIOx_MODER is used to select the I/O mode (input, output, AF or analog).

GPIOx_OTYPER and GPIOx_OSPEEDR are used to select the output type (push-pull or open-drain) and speed.

GPIOx_PUPDR is used to select the pull-up/pull-down whatever the I/O direction.

8.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped input and output data registers: GPIOx_IDR and GPIOx_ODR.

GPIOx_ODR stores the data to be output, it is read/write accessible.

Data input through the I/O are stored into GPIOx_IDR, it is a read-only register.

8.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). GPIOx_BSRR has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR correspond two control bits in GPIOx_BSRR, BS(i) and BR(i):

- When written to 1, BS(i) sets the corresponding ODR(i) bit.
- When written to 1, BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using GPIOx_BSRR to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. GPIOx_BSRR provides a way of performing atomic bitwise handling.

There is no software need to disable interrupts when programming GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

8.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. Frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write GPIOx_LCKR, a specific write/read sequence must be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the I/Os configuration (during the write sequence, LCKR[15:0] value must be the same).

When the LOCK sequence is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset.

Each GPIOx_LCKR bit freezes the corresponding bit in the control registers GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence can only be performed using a word (32-bit long) access to GPIOx_LCKR due to the fact that GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

8.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O: GPIOx_AFRL and GPIOx_AFRH.

With these registers, the user can connect an alternate function to some other pin as required by the application. This means that a number of possible peripheral functions are multiplexed on each GPIO. The application can thus select any one of the possible functions for each I/O.

The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the product datasheet.

8.3.8 External interrupt/wakeup lines

All ports have external interrupt capability.

To use external interrupt lines, the port must be configured in input mode.

Refer to *Section 14.4.1: EXTI configurable event input wakeup*.

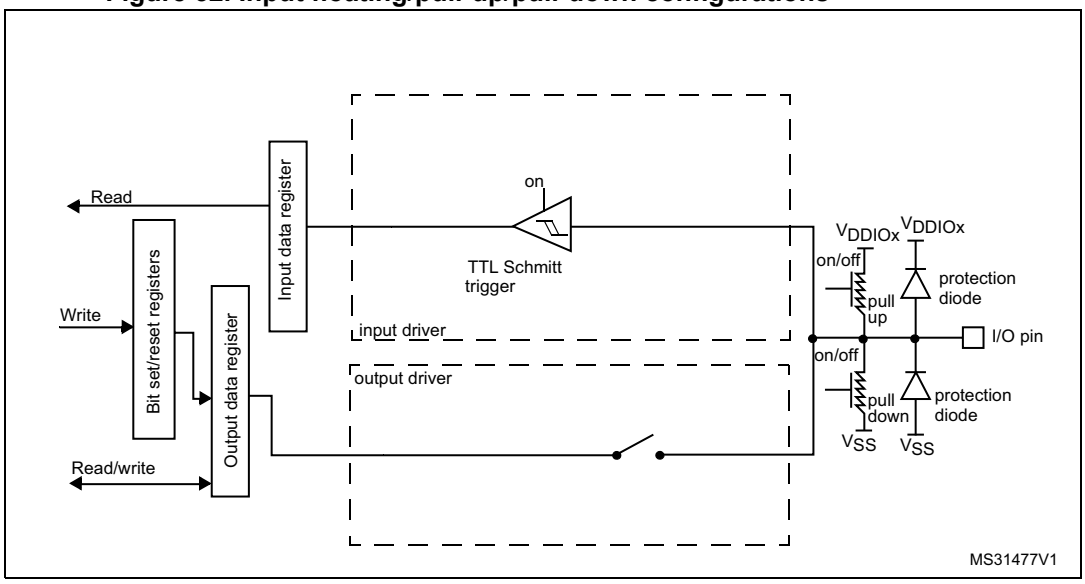
8.3.9 Input configuration

When the I/O port is programmed as input, the following occurs:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

The figure below shows the input configuration of the I/O port bit.

Figure 32. Input floating/pull-up/pull-down configurations



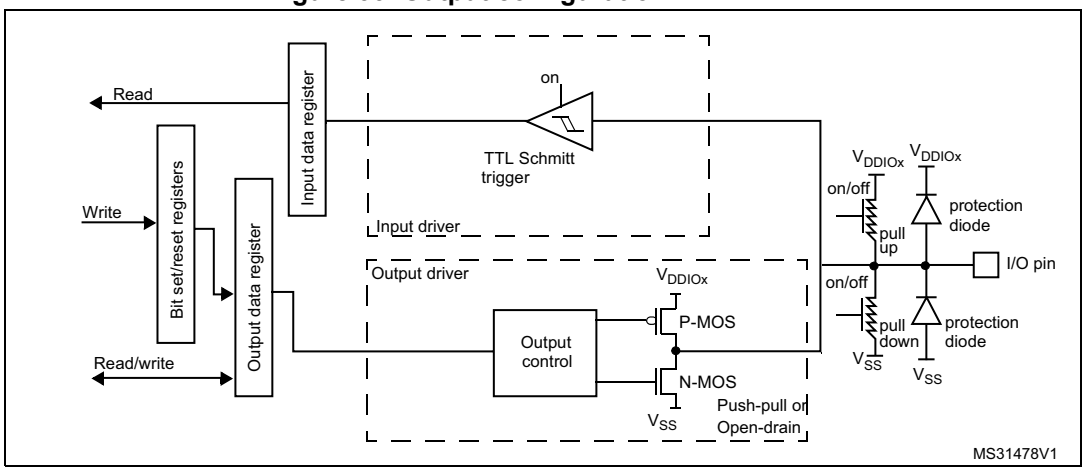
8.3.10 Output configuration

When the I/O port is programmed as output, the following occurs:

- The output buffer is enabled:
  - Open drain mode: A 0 in the Output register activates the N-MOS whereas a 1 in the Output register leaves the port in Hi-Z (the P-MOS is never activated).
  - Push-pull mode: A 0 in the Output register activates the N-MOS whereas a 1 in the Output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

The figure below shows the output configuration of the I/O port bit.

Figure 33. Output configuration



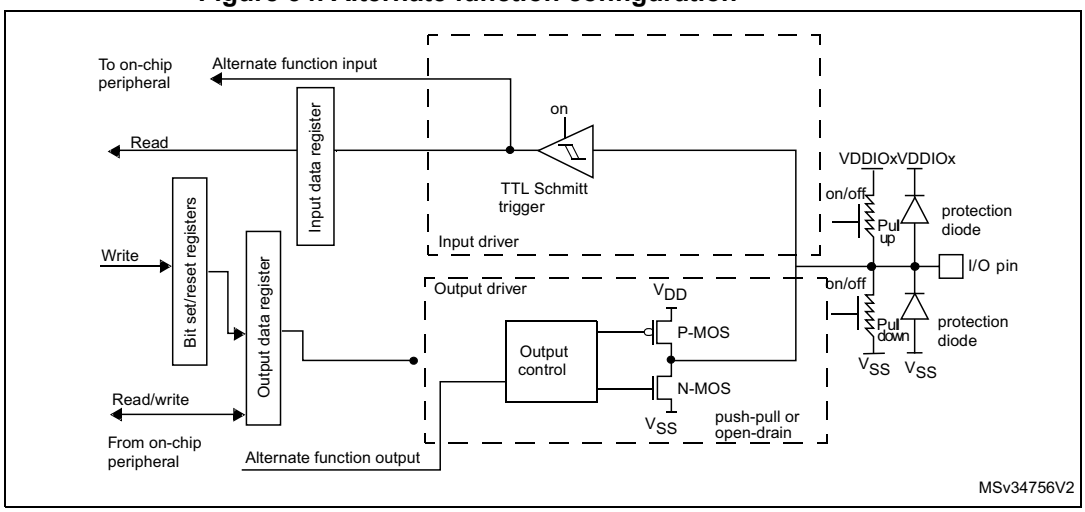
8.3.11 Alternate function configuration

When the I/O port is programmed as alternate function, the following occurs:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.

The figure below shows the alternate function configuration of the I/O port bit.

Figure 34. Alternate function configuration



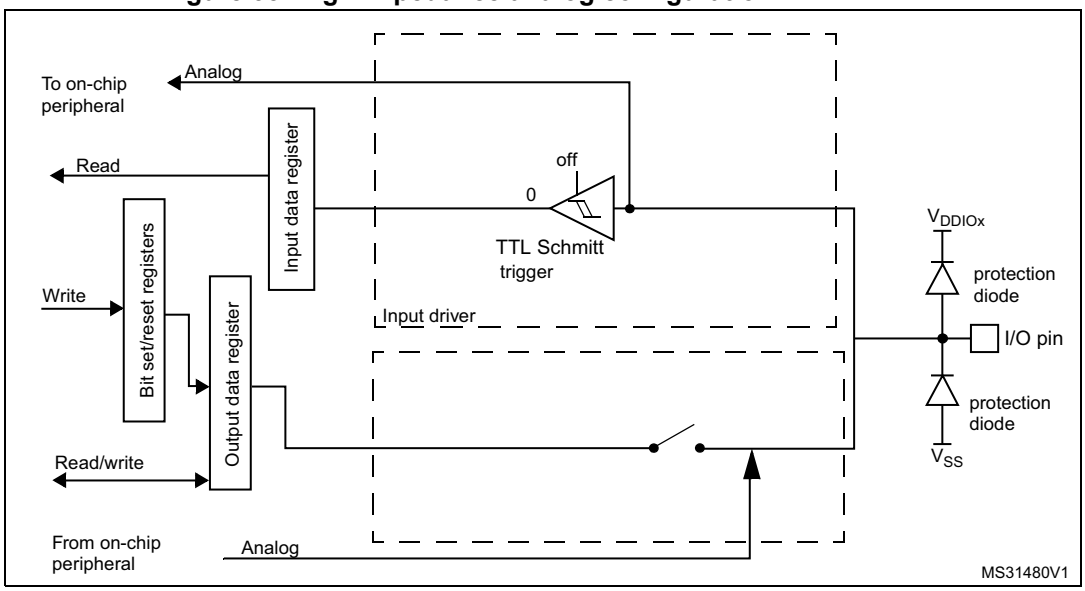
8.3.12 Analog configuration

When the I/O port is programmed as analog configuration, the following occurs:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled by hardware.
- Read access to the input data register gets the value 0.

The figure below shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 35. High impedance analog configuration



8.3.13 Using the LSE oscillator pins as GPIOs

When the LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the LSE oscillator is switched on (by setting the LSEON bit in the RCC_CSR register), the oscillator takes the control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the OSC32_IN pin is reserved for clock input and the OSC32_OUT pin can still be used as normal GPIO.

Note: The HSE32 OSC_IN and OSC_OUT pins are dedicated oscillator pins and cannot be used as GPIO.

8.3.14 Using the GPIO pins in the RTC supply domain

GPIO functionality of PC13, PC14 and PC15 is lost when the core supply domain is powered off (device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to *Section 30: Real-time clock (RTC)*.

8.3.15 Using PH3 as GPIO

PH3 may be used as boot pin (BOOT0) or as GPIO.

PH3 switches from the input mode to the analog input mode depending on the nSWBOOT0 bit in the user option byte as follows:

- After the option byte loading phase if nSWBOOT0 = 1.
- After reset if nSWBOOT0 = 0.

8.4 GPIO registers

This section gives a detailed description of the GPIO registers for GPIOx port, with x = A to C and x = H.

For a summary of register bits, register address offsets and reset values, refer to *Table 60* to *Table 63*.

The peripheral registers can be written in word, half word or byte mode.

8.4.1 GPIOx mode register (GPIOx_MODER) (x = A to B)

Address offset: Block A: 0x0000

Address offset: Block B: 0x0400

Reset value: Block A: 0xABFF FFFF

Reset value: Block B: 0xFFFF FEBF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port Pxy I/O type configuration (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

8.4.2 GPIOx output type register (GPIOx_OTYPER) (x = A to B)

Address offset: Block A: 0x0004

Address offset: Block B: 0x0404

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port Pxy output type configuration (y = 15 to 0)
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

8.4.3 GPIOx output speed register (GPIOx_OSPEEDR) (x = A to B)

Address offset: Block A: 0x0008

Address offset: Block B: 0x0408

Reset value: Block A: 0x0C00 0000

Reset value: Block B: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDy[1:0]**: Port Pxy output speed configuration (y = 15 to 0)
These bits are written by software to configure the I/O output speed.
00: Low speed
01: Medium speed
10: Fast speed
11: High speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

8.4.4 GPIOx pull-up/pull-down register (GPIOx_PUPDR) (x = A to B)

Address offset: Block A: 0x000C

Address offset: Block B: 0x040C

Reset value: Block A: 0x6400 0000

Reset value: Block B: 0x0000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDy[1:0]:** Port Pxy pull configuration (y = 15 to 0)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

8.4.5 GPIOx input data register (GPIOx_IDR) (x = A to B)

Address offset: Block A: 0x0010

Address offset: Block B: 0x0410

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port Pxy input data bit (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

8.4.6 GPIOx output data register (GPIOx_ODR) (x = A to B)

Address offset: Block A: 0x0014

Address offset: Block B: 0x0414

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port Pxy output data (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, OD bits can be individually set and/or reset by writing to the GPIOx_BSRR and GPIOx_BRR registers.

8.4.7 GPIOx bit set/reset register (GPIOx_BSRR) (x = A to B)

Address offset: Block A: 0x0018

Address offset: Block B: 0x0418

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 **BRy:** Port Pxy reset output data bit [15:0] in GPIOx_ODR (y = 15 to 0)

These bits are read clear-write 1. A read to these bits returns the value 0x0000.

0: No action on the corresponding GPIOx_ODR.OD[y] bit

1: Resets the corresponding GPIOx_ODR.OD[y] bit.

Note: If both BSy and BRy are set, BSy has priority.

Bits 15:0 **BSy:** Port Pxy set output data bit [15:0] in GPIOx_ODR (y = 15 to 0)

These bits are read clear-write 1. A read to these bits returns the value 0x0000.

0: No action on the corresponding GPIOx_ODR.OD[y] bit

1: Sets the corresponding GPIOx_ODR.OD[y] bit.

8.4.8 GPIOx configuration lock register (GPIOx_LCKR) (x = A to B)

Address offset: Block A: 0x001C

Address offset: Block B: 0x041C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the lock sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to GPIOx_LCKR. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = 1 + LCKR[15:0]

WR LCKR[16] = 0 + LCKR[15:0]

WR LCKR[16] = 1 + LCKR[15:0]

RD LCKR

RD LCKR[16] = 1 (This read operation is optional but it confirms that the lock is active.)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port Px[15:0] lock configuration (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0.

0: Port Pxy configuration not locked

1: Port Pxy configuration locked

8.4.9 GPIOx alternate function low register (GPIOx_AFRL) (x = A to B)

Address offset: Block A: 0x0020

Address offset: Block B: 0x0420

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Port Pxy alternate function selection (y = 7 to 0)
These bits are written by software to configure alternate function I/Os
0x0: AF0 selected
0x1: AF1 selected
0x2: AF2 selected
...
0xE: AF14 selected
0xF: AF15 selected

8.4.10 GPIOx alternate function high register (GPIOx_AFRH) (x = A to B)

Address offset: Block A: 0x0024

Address offset: Block B: 0x0424

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Port Pxy alternate function selection (y = 15 to 8)
These bits are written by software to configure alternate function I/Os
0x0: AF0 selected
0x1: AF1 selected
0x2: AF2 selected
...
0xE: AF14 selected
0xF: AF15 selected

8.4.11 GPIOx bit reset register (GPIOx_BRR) (x = A to B)

Address offset: Block A: 0x0028

Address offset: Block B: 0x0428

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRy:** Port Pxy reset output data bit [15:0] in GPIOx_ODR (y = 15 to 0)
These bits are read clear-write 1. A read to these bits returns the value 0x0000.
0: No action on the corresponding GPIOx_ODR.OD[y] bit
1: Resets the corresponding GPIOx_ODR.OD[y] bit.

8.4.12 GPIOC mode register (GPIOC_MODER)

Address offset: 0x0800

Reset value: 0xFC00 3FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **MODE15[1:0]:** Port PC15 IO type configuration

Bits 29:28 **MODE14[1:0]:** Port PC14 IO type configuration

Bits 27:26 **MODE13[1:0]:** Port PC13 IO type configuration

Bits 25:14 Reserved, must be kept at reset value.

Bits 13:12 **MODE6[1:0]:** Port PC6 IO type configuration

Bits 11:10 **MODE5[1:0]:** Port PC5 IO type configuration

Bits 9:8 **MODE4[1:0]:** Port PC4 IO type configuration

Bits 7:6 **MODE3[1:0]:** Port PC3 IO type configuration

Bits 5:4 **MODE2[1:0]:** Port PC2 IO type configuration

Bits 3:2 **MODE1[1:0]:** Port PC1 IO type configuration

Bits 1:0 **MODE0[1:0]:** Port PC0 IO type configuration

These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

8.4.13 GPIOC output type register (GPIOC_OTYPER)

Address offset: 0x0804

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	Res.	Res.	Res.	Res.	Res.	Res.	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **OTy:** Port PCy output type configuration (y = 15 to 13)

These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

Bits 12:7 Reserved, must be kept at reset value.

Bits 6:0 **OTy:** Port PCy output type configuration (y = 6 to 0)

8.4.14 GPIOC output speed register (GPIOC_OSPEEDR)

Address offset: 0x0808

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **OSPEED15[1:0]**: Port PC15 output speed configuration

Bits 29:28 **OSPEED14[1:0]**: Port PC14 output speed configuration

Bits 27:26 **OSPEED13[1:0]**: Port PC13 output speed configuration

Bits 25:14 Reserved, must be kept at reset value.

Bits 13:12 **OSPEED6[1:0]**: Port PC6 output speed configuration

Bits 11:10 **OSPEED5[1:0]**: Port PC5 output speed configuration

Bits 9:8 **OSPEED4[1:0]**: Port PC4 output speed configuration

Bits 7:6 **OSPEED3[1:0]**: Port PC3 output speed configuration

Bits 5:4 **OSPEED2[1:0]**: Port PC2 output speed configuration

Bits 3:2 **OSPEED1[1:0]**: Port PC1 output speed configuration

Bits 1:0 **OSPEED0[1:0]**: Port PC0 output speed configuration

These bits are written by software to configure the I/O output speed.
00: Low speed
01: Medium speed
10: Fast speed
11: High speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

8.4.15 GPIOC pull-up/pull-down register (GPIOC_PUPDR)

Address offset: 0x080C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **PUPD15[1:0]:** Port PC15 pull configuration

Bits 29:28 **PUPD14[1:0]:** Port PC14 pull configuration

Bits 27:26 **PUPD13[1:0]:** Port PC13 pull configuration

Bits 25:14 Reserved, must be kept at reset value.

Bits 13:12 **PUPD6[1:0]:** Port PC6 pull configuration

Bits 11:10 **PUPD5[1:0]:** Port PC5 pull configuration

Bits 9:8 **PUPD4[1:0]:** Port PC4 pull configuration

Bits 7:6 **PUPD3[1:0]:** Port PC3 pull configuration

Bits 5:4 **PUPD2[1:0]:** Port PC2 pull configuration

Bits 3:2 **PUPD1[1:0]:** Port PC1 pull configuration

Bits 1:0 **PUPD0[1:0]:** Port PC0 pull configuration

These bits are written by software to configure the I/O pull-up or pull-down.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

8.4.16 GPIOC input data register (GPIOC_IDR)

Address offset: 0x0810

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	Res.	Res.	Res.	Res.	Res.	Res.	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r							r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **IDy:** Port PCy input data bit (y = 15 to 13)
These bits are read-only. They contain the input value of the corresponding I/O port.

Bits 12:7 Reserved, must be kept at reset value.

Bits 6:0 **IDy:** Port PCy input data bit (y = 6 to 0)

8.4.17 GPIOC output data register (GPIOC_ODR)

Address offset: 0x0814

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	Res.	Res.	Res.	Res.	Res.	Res.	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **ODy:** Port PCy output data bit (y = 15 to 13)
These bits can be read and written by software.

Note: For atomic bit set/reset, OD bits can be individually set and/or reset by writing to the GPIOC_BSRR and GPIOC_BRR registers.

Bits 12:7 Reserved, must be kept at reset value.

Bits 6:0 **ODy:** Port PCy output data bit (y = 6 to 0)

8.4.18 GPIOC bit set/reset register (GPIOC_BSRR)

Address offset: 0x0818

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	BR6	BR5	BR4	BR3	BR2	BR1	BR0
rc_w1	rc_w1	rc_w1							rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	Res.	Res.	Res.	Res.	Res.	Res.	BS6	BS5	BS4	BS3	BS2	BS1	BS0
rc_w1	rc_w1	rc_w1							rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:29 **BRy:** Port PCy reset output data bit [y] in GPIOC_ODR (y = 15 to 13)
These bits are read clear-write 1. A read to these bits returns the value 0.
0: No action on the corresponding GPIOC_ODR.OD0
1: Resets the corresponding GPIOC_ODR.OD0.
Note: If both BS0 and BR0 are set, BS0 has priority.

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:16 **BRy:** Port PCy reset output data bit [y] in GPIOC_ODR (y = 6 to 0)

Bits 15:13 **BSy:** Port PCy set output data bit [y] in GPIOC_ODR (y = 15 to 13)
These bits are read clear-write 1. A read to these bits returns the value 0.
0: No action on the corresponding GPIOC_ODR.OD0
1: Resets the corresponding GPIOC_ODR.OD0.
Note: If both BS0 and BR0 are set, BS0 has priority.

Bits 12:7 Reserved, must be kept at reset value.

Bits 6:0 **BSy:** Port PCy set output data bit [y] in GPIOC_ODR (y = 6 to 0)

8.4.19 GPIOC configuration lock register (GPIOC_LCKR)

Address offset: 0x081C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the lock sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOC_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	Res.	Res.	Res.	Res.	Res.	Res.	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.
0: Port PC configuration lock key not active
1: Port PC configuration lock key active. GPIOC_LCKR is locked until the next MCU reset or peripheral reset.
LOCK key write sequence:
WR LCKR[16] = 1 + LCKR[15:0]
WR LCKR[16] = 0 + LCKR[15:0]
WR LCKR[16] = 1 + LCKR[15:0]
RD LCKR
RD LCKR[16] = 1 (This read operation is optional but it confirms that the lock is active.)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.
Any error in the lock sequence aborts the lock.
After the first lock sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:13 **LCKy:** Port PCy lock configuration (y = 15 to 13)

This bit is read/write but can only be written when the LCKK bit is 0.
0: Port PCy configuration not locked
1: Port PCy configuration locked

Bits 12:7 Reserved, must be kept at reset value.

Bits 6:0 **LCKy:** Port PCy lock configuration (y = 6 to 0)

8.4.20 GPIOC alternate function low register (GPIOC_AFRL)

Address offset: 0x0820

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **AFSEL6[3:0]:** Port PC6 alternate function selection

Bits 23:20 **AFSEL5[3:0]:** Port PC5 alternate function selection

Bits 19:16 **AFSEL4[3:0]:** Port PC4 alternate function selection

Bits 15:12 **AFSEL3[3:0]:** Port PC3 alternate function selection

Bits 11:8 **AFSEL2[3:0]:** Port PC2 alternate function selection

Bits 7:4 **AFSEL1[3:0]:** Port PC1 alternate function selection

Bits 3:0 **AFSEL0[3:0]:** Port PC0 alternate function selection

These bits are written by software to configure alternate function I/Os.
0x0: AF0 selected
0x1: AF1 selected
0x2: AF2 selected
...
0xE: AF14 selected
0xF: AF15 selected

8.4.21 GPIOC alternate function high register (GPIOC_AFRH)

Address offset: 0x0824

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 **AFSEL15[3:0]:** Port PC15 alternate function selection

Bits 27:24 **AFSEL14[3:0]:** Port PC14 alternate function selection

Bits 23:20 **AFSEL13[3:0]:** Port PC13 alternate function selection
These bits are written by software to configure alternate function I/Os.
0x0: AF0 selected
0x1: AF1 selected
0x2: AF2 selected
...
0xE: AF14 selected
0xF: AF15 selected

Bits 19:0 Reserved, must be kept at reset value.

8.4.22 GPIOC bit reset register (GPIOC_BRR)

Address offset: 0x0828

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	BR6	BR5	BR4	BR3	BR2	BR1	BR0
rc_w1	rc_w1	rc_w1							rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **BRy:** Port PCy reset output data bit [15] in GPIOC_ODR (y = 15 to 13)
These bits are read clear-write 1. A read to this bit returns the value 0.
0: No action on the corresponding GPIOC_ODR.OD0
1: Resets the corresponding GPIOC_ODR.OD0.

Bits 12:7 Reserved, must be kept at reset value.

Bits 6:0 **BRy:** Port PCy reset output data bit [6] in GPIOC_ODR (y = 6 to 0)

8.4.23 GPIOH mode register (GPIOH_MODER)

Address offset: 0x1C00

Reset value: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **MODE3[1:0]:** Port PH3 IO type configuration
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Bits 5:0 Reserved, must be kept at reset value.

8.4.24 GPIO H output type register (GPIOH_OTYPER)

Address offset: 0x1C04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **OT3:** Port PH3 output type configuration
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

Bits 2:0 Reserved, must be kept at reset value.

8.4.25 GPIOH output speed register (GPIOH_OSPEEDR)

Address offset: 0x1C08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **OSPEED3[1:0]**: Port PH3 output speed configuration
These bits are written by software to configure the I/O output speed.
00: Low speed
01: Medium speed
10: Fast speed
11: High speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Bits 5:0 Reserved, must be kept at reset value.

8.4.26 GPIOH pull-up/pull-down register (GPIOH_PUPDR)

Address offset: 0x1C0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPD3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **PUPD3[1:0]:** Port PH3 pull configuration
These bits are written by software to configure the I/O pull-up or pull-down.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Bits 5:0 Reserved, must be kept at reset value.

8.4.27 GPIOH input data register (GPIOH_IDR)

Address offset: 0x1C10

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID3	Res.	Res.	Res.
												r			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ID3:** Port PH3 input data bit
This bit is read-only. It contains the input value of the corresponding I/O port.

Bits 2:0 Reserved, must be kept at reset value.

8.4.28 GPIOH output data register (GPIOH_ODR)

Address offset: 0x1C14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **OD3:** Port PH3 output data

This bit can be read and written by software.

Note: For atomic bit set/reset, OD bits can be individually set and/or reset by writing to the GPIOH_BSRR and GPIOH_BRR registers.

Bits 2:0 Reserved, must be kept at reset value.

8.4.29 GPIO H bit set/reset register (GPIOH_BSRR)

Address offset: 0x1C18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.
												rc_w1			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BS3	Res.	Res.	Res.
												rc_w1			

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **BR3:** Port PH3 reset output data bit [3] in GPIOH_ODR

This bit is read clear-write 1. A read to this bit returns the value 0.

0: No action on the corresponding GPIOH_ODR.OD3

1: Resets the corresponding GPIOH_ODR.OD3.

Note: If both BS3 and BR3 are set, BS3 has priority.

Bits 18:4 Reserved, must be kept at reset value.

Bit 3 **BS3:** Port PH3 set output data bit [3] in GPIOH_ODR

This bit is read clear-write 1. A read to this bit returns the value 0.

0: No action on the corresponding GPIOH_ODR.OD3

1: Sets the corresponding GPIOH_ODR.OD3.

Bits 2:0 Reserved, must be kept at reset value.

8.4.30 GPIOH configuration lock register (GPIOH_LCKR)

Address offset: 0x1C1C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the lock sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOH_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK3	Res.	Res.	Res.
												rw			

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.
0: Port PH configuration lock key not active
1: Port PH configuration lock key active. GPIOH_LCKR is locked until the next MCU reset or peripheral reset.
LOCK key write sequence:
WR LCKR[16] = 1 + LCKR[15:0]
WR LCKR[16] = 0 + LCKR[15:0]
WR LCKR[16] = 1 + LCKR[15:0]
RD LCKR
RD LCKR[16] = 1 (This read operation is optional but it confirms that the lock is active.)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.
Any error in the lock sequence aborts the lock.
After the first lock sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **LCK3:** Port PH3 lock configuration

This bit is read/write but can only be written when the LCKK bit is 0.
0: Port PH3 configuration not locked
1: Port PH3 configuration locked

Bits 2:0 Reserved, must be kept at reset value.

8.4.31 GPIOH alternate function low register (GPIOH_AFRL)

Address offset: 0x1C20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **AFSEL3[3:0]:** Port PH3 alternate function selection
These bits are written by software to configure alternate function I/Os.
0x0: AF0 selected
0x1: AF1 selected
0x2: AF2 selected
...
0xE: AF14 selected
0xF: AF15 selected

Bits 11:0 Reserved, must be kept at reset value.

8.4.32 GPIOH bit reset register (GPIOH_BRR)

Address offset: 0x1C28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.
												rc_w1			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BR3:** Port PH3 reset output data bit [3] in GPIOH_ODR
This bit is read clear-write 1. A read to this bit returns the value 0.
0: No action on the corresponding GPIOH_ODR.OD3
1: Resets the corresponding GPIOH_ODR.OD3.

Bits 2:0 Reserved, must be kept at reset value.

8.4.33 GPIOA register map

Table 60. GPIOA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	GPIOA_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0004	GPIOA_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	GPIOA_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	GPIOA_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0010	GPIOA_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0014	GPIOA_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	GPIOA_BSRR	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	GPIOA_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	GPIOA_AFRL	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	GPIOA_AFRH	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	GPIOA_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

8.4.34 GPIOB register map

Table 61. GPIOB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0400	GPIOB_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
0x0404	GPIOB_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0408	GPIOB_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x040C	GPIOB_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x0410	GPIOB_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0414	GPIOB_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0418	GPIOB_BSRR	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x041C	GPIOB_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0420	GPIOB_AFRL	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0424	GPIOB_AFRH	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0428	GPIOB_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

8.4.35 GPIOC register map

Table 62. GPIOC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0800	GPIOC_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	1	1	1	1	1													1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0804	GPIOC_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	Res.	Res.	Res.	Res.	Res.	Res.	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0							0	0	0	0	0	0	0
0x0808	GPIOC_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080C	GPIOC_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0810	GPIOC_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	Res.	Res.	Res.	Res.	Res.	Res.	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x							x	x	x	x	x	x	x
0x0814	GPIOC_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	Res.	Res.	Res.	Res.	Res.	Res.	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0							0	0	0	0	0	0	0
0x0818	GPIOC_BSRR	BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	Res.	Res.	Res.	Res.	Res.	Res.	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0
0x081C	GPIOC_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	Res.	Res.	Res.	Res.	Res.	Res.	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0							0	0	0	0	0	0	0
0x0820	GPIOC_AFRL	Res.	Res.	Res.	Res.	AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]	
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0824	GPIOC_AFRH	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0																				
0x0828	GPIOC_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0							0	0	0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

8.4.36 GPIOH register map

Table 63. GPIOH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C00	GPIOH_MODER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									1	1						
0x1C04	GPIOH_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT3	Res.	Res.	Res.
	Reset value																													0			
0x1C08	GPIOH_OSPEEDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									0	0						
0x1C0C	GPIOH_PUPDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPD3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									0	0						
0x1C10	GPIOH_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID3	Res.	Res.	Res.
	Reset value																													x			
0x1C14	GPIOH_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD3	Res.	Res.	Res.
	Reset value																													0			
0x1C18	GPIOH_BSRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BS3	Res.	Res.	Res.
	Reset value													0																0			
0x1C1C	GPIOH_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK3	Res.	Res.	Res.
	Reset value																0													0			
0x1C20	GPIOH_AFRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL3[3:0]															
	Reset value																	0	0	0	0												
0x1C24	Reserved	Reserved.																															
0x1C28	GPIOH_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.
	Reset value																													0			

Refer to *Section 2.4* for the register boundary addresses.

9 System configuration controller (SYSCFG)

9.1 SYSCFG main features

STM32WLEx devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Remapping memory areas
- Managing the external interrupt line connection to the GPIOs
- Managing robustness feature
- Setting SRAM2 write protection and SRAM2 software erase
- SRAM1, SRAM2 and PKA SRAM erase busy flags
- Enabling /disabling I^2C Fast-mode Plus driving capability on some I/Os and voltage booster for I/Os analog switches.

9.2 SYSCFG registers

9.2.1 SYSCFG memory remap register (SYSCFG_MEMRMP)

This register is used for specific configurations on memory remap.

Address offset: 0x000

Reset value: 0x0000 000X

MEM_MODE[2:0] is the memory mode selected by the BOOT0 pin and BOOT1 option bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM_MODE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **MEM_MODE[2:0]:** memory mapping selection

These bits control the memory internal mapping at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT mode setting. After reset, these bits take the value selected by BOOT0 (pin or option bit depending on nSWBOOT0 option bit) and BOOT1 option bit.

000: Main Flash memory mapped at CPU 0x00000000

001: System Flash memory mapped at CPU 0x00000000

010: Reserved

011: SRAM1 mapped at CPU 0x00000000

100: Reserved

101: Reserved

110: Reserved

111: Reserved

9.2.2 SYSCFG configuration register 1 (SYSCFG_CFGR1)

Address offset: 0x004

Reset value: 0x7C00 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C3_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **I2C3_FMP:** I2C3 Fast-mode Plus driving capability activation
This bit enables the Fm+ driving mode on I2C3 pins selected through AF selection bits.
0: Fm+ mode is not enabled on I2C3 pins selected through AF selection bits
1: Fm+ mode is enabled on I2C3 pins selected through AF selection bits.

Bit 21 **I2C2_FMP:** I2C2 Fast-mode Plus driving capability activation
This bit enables the Fm+ driving mode on I2C2 pins selected through AF selection bits.
0: Fm+ mode is not enabled on I2C2 pins selected through AF selection bits
1: Fm+ mode is enabled on I2C2 pins selected through AF selection bits.

Bit 20 **I2C1_FMP:** I2C1 Fast-mode Plus driving capability activation
This bit enables the Fm+ driving mode on I2C1 pins selected through AF selection bits.
0: Fm+ mode is not enabled on I2C1 pins selected through AF selection bits
1: Fm+ mode is enabled on I2C1 pins selected through AF selection bits.

Bit 19 **I2C_PB9_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB9
This bit enables the Fm+ driving mode for PB9.
0: PB9 pin operates in standard mode.
1: Fm+ mode enabled on PB9 pin, and the speed control is bypassed.

Bit 18 **I2C_PB8_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB8
This bit enables the Fm+ driving mode for PB8.
0: PB8 pin operates in standard mode.
1: Fm+ mode enabled on PB8 pin, and the speed control is bypassed.

Bit 17 **I2C_PB7_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB7
This bit enables the Fm+ driving mode for PB7.
0: PB7 pin operates in standard mode.
1: Fm+ mode enabled on PB7 pin, and the speed control is bypassed.

Bit 16 **I2C_PB6_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB6
This bit enables the Fm+ driving mode for PB6.
0: PB6 pin operates in standard mode.
1: Fm+ mode enabled on PB6 pin, and the Speed control is bypassed.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BOOSTEN:** I/O analog switch voltage booster enable

0: I/O analog switches are supplied by V_{DDA} voltage. This is the recommended configuration when using the ADC in high V_{DDA} voltage operation.

1: I/O analog switches are supplied by a dedicated voltage booster (supplied by V_{DD}). This is the recommended configuration when using the ADC in low V_{DDA} voltage operation.

Bits 7:0 Reserved, must be kept at reset value.

9.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EXTI3[2:0]			Res.	EXTI2[2:0]			Res.	EXTI1[2:0]			Res.	EXTI0[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **EXTI3[2:0]**: EXTI3 configuration bits

These bits are written by software to select the source input for the EXTI3 external interrupt.
000: PA3 pin
001: PB3 pin
010: PC3 pin
111: PH3 pin
Others: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI2[2:0]**: EXTI2 configuration bits

These bits are written by software to select the source input for the EXTI2 external interrupt.
000: PA2 pin
001: PB2 pin
010: PC2 pin
Others: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI1[2:0]**: EXTI1 configuration bits

These bits are written by software to select the source input for the EXTI1 external interrupt.
000: PA1 pin
001: PB1 pin
010: PC1 pin
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI0[2:0]**: EXTI0 configuration bits
These bits are written by software to select the source input for the EXTI0 external interrupt.
000: PA0 pin
001: PB0 pin
010: PC0 pin
Others: Reserved

Note: Some of the I/O pins mentioned in this register may not be available on small packages.

9.2.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EXTI7[2:0]			Res.	EXTI6[2:0]			Res.	EXTI5[2:0]			Res.	EXTI4[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **EXTI7[2:0]**: EXTI7 configuration bits
These bits are written by software to select the source input for the EXTI7 external interrupt.
000: PA7 pin
001: PB7 pin
Others: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI6[2:0]**: EXTI6 configuration bits
These bits are written by software to select the source input for the EXTI6 external interrupt.
000: PA6 pin
001: PB6 pin
010: PC6 pin
0Others: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI5[2:0]**: EXTI5 configuration bits
These bits are written by software to select the source input for the EXTI5 external interrupt.
000: PA5 pin
001: PB5 pin
010: PC5 pin
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI4[2:0]**: EXTI4 configuration bits
These bits are written by software to select the source input for the EXTI4 external interrupt.
000: PA4 pin
001: PB4 pin
010: PC4 pin
Others: Reserved

Note: Some of the I/O pins mentioned in this register may not be available on small packages.

9.2.5 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EXTI11[2:0]			Res.	EXTI10[2:0]			Res.	EXTI9[2:0]			Res.	EXTI8[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **EXTI11[2:0]**: EXTI11 configuration bits
These bits are written by software to select the source input for the EXTI11 external interrupt.
000: PA11 pin
001: PB11 pin
Others: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI10[2:0]**: EXTI10 configuration bits
These bits are written by software to select the source input for the EXTI10 external interrupt.
000: PA10 pin
001: PB10 pin
Others: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI9[2:0]**: EXTI9 configuration bits
These bits are written by software to select the source input for the EXTI9 external interrupt.
000: PA9 pin
001: PB9 pin
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI8[2:0]**: EXTI8 configuration bits
These bits are written by software to select the source input for the EXTI8 external interrupt.
000: PA8 pin
001: PB8 pin
Others: Reserved

Note: Some of the I/O pins mentioned in this register may not be available on small packages.

9.2.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EXTI15[2:0]			Res.	EXTI14[2:0]			Res.	EXTI13[2:0]			Res.	EXTI12[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **EXTI15[2:0]**: EXTI15 configuration bits
These bits are written by software to select the source input for the EXTI15 external interrupt.
000: PA15 pin
001: PB15 pin
010: PC15 pin
Others: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI14[2:0]**: EXTI14 configuration bits
These bits are written by software to select the source input for the EXTI14 external interrupt.
000: PA14 pin
001: PB14 pin
010: PC14 pin
Others: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI13[2:0]**: EXTI13 configuration bits
These bits are written by software to select the source input for the EXTI13 external interrupt.
000: PA13 pin
001: PB13 pin
010: PC13 pin
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI12[2:0]**: EXTI12 configuration bits
These bits are written by software to select the source input for the EXTI12 external interrupt.
000: PA12 pin
001: PB12 pin
Others: Reserved

Note: Some of the I/O pins mentioned in this register may not be available on small packages.

9.2.7 SYSCFG SRAM control and status register (SYSCFG_SCSR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKASRAMBSY	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBSY	SRAM2ER
							r							r	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PKASRAMBSY:** PKA SRAM busy by erase operation
0: No PKA SRAM erase operation is ongoing.
1: PKA SRAM erase operation is ongoing.
See *Section 2.3: SRAM erase* for more information on SRAM erase conditions

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **SRAMBSY:** SRAM1 or SRAM2 busy by erase operation
0: No SRAM1 or SRAM2 erase operation is ongoing.
1: SRAM1 or SRAM2 erase operation is ongoing.
See *Section 2.3: SRAM erase* for more information on SRAM erase conditions

Bit 0 **SRAM2ER:** SRAM2 erase
Setting this bit starts a hardware SRAM2 erase operation. This bit is automatically cleared at the end of the SRAM2 erase operation.
Note: This bit is write-protected: setting this bit is possible only after the correct key sequence is written in the SYSCFG_SKR register.

9.2.8 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPF	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL
							rc_w1					rs	rs	rs	rs

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SPF**: SRAM2 parity error flag
This bit is set by hardware when an SRAM2 parity error is detected. It is cleared by software by writing ‘1’.
0: No SRAM2 parity error detected
1: SRAM2 parity error detected

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ECCL**: ECC lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Flash ECC error connection to TIM1/16/17 break input.

0: ECC error disconnected from TIM1/16/17 break input.

1: ECC error connected to TIM1/16/17 break input.

Bit 2 **PVDL**: PVD lock enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/16/17 break input, as well as the PVDE and PLS[2:0] in the PWR_CR2R register.

0: PVD interrupt disconnected from TIM1/16/17 break input. PVDE and PLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/16/17 break input. PVDE and PLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM2 parity lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM2 parity error signal connection to TIM1/16/17 break inputs.

0: SRAM2 parity error signal disconnected from TIM1/16/17 break inputs

1: SRAM2 parity error signal connected to TIM1/16/17 break inputs

Bit 0 **CLL:** CPU LOCKUP (Hardfault) output enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the connection of CPU LOCKUP (Hardfault) output to TIM1/16/17 break inputs.

0: CPU LOCKUP output disconnected from TIM1/16/17 break inputs

1: CPU LOCKUP output connected to TIM1/16/17 break inputs

9.2.9 SYSCFG SRAM2 write protection register (SYSCFG_SWPR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PxWP**: SRAM2 1 Kbyte page x write protection (x = 31 to 0)

These bits are set by software and cleared only by a system reset. Number of pages depend on SRAM2 size available from the device type, see data sheet.

0: Write protection of SRAM2 1 Kbyte page x is disabled.

1: Write protection of SRAM2 1 Kbyte page x is enabled.

9.2.10 SYSCFG SRAM2 key register (SYSCFG_SKR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: SRAM2 write protection key for software erase

The following steps are required to unlock the write protection of the SRAM2ER bit in the SYSCFG_SCSR register.

1. Write 0xCA into Key[7:0].
2. Write 0x53 into Key[7:0].

Writing a wrong key reactivates the write protection.

9.2.11 SYSCFG radio debug control register (SYSCFG_RFDCR)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFTBSEL
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **RFTBSEL**: radio debug test bus selection

0 Digital test bus selected on RF_ADTB[3:0]

1: Analog test bus selected on RF_ADTB[3:0]

9.2.12 SYSCFG register map

Table 64. SYSCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SYSCFG_ MEMRMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM_MODE [2:0]		
	Reset value																														x	x	x

Table 64. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	SYSCFG_CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C3_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0	0	0	0	0								0								
0x008	SYSCFG_EXTICR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI3 [2:0]			Res.	EXTI2 [2:0]			Res.	EXTI1 [2:0]			Res.	EXTI0 [2:0]		
	Reset value																		0	0	0		0	0	0		0	0	0		0	0	0
0x00C	SYSCFG_EXTICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI7 [2:0]			Res.	EXTI6 [2:0]			Res.	EXTI5 [2:0]			Res.	EXTI4 [2:0]		
	Reset value																		0	0	0		0	0	0		0	0	0		0	0	0
0x010	SYSCFG_EXTICR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI11 [2:0]			Res.	EXTI10 [2:0]			Res.	EXTI9 [2:0]			Res.	EXTI8 [2:0]		
	Reset value																		0	0	0		0	0	0		0	0	0		0	0	0
0x014	SYSCFG_EXTICR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI15 [2:0]			Res.	EXTI14 [2:0]			Res.	EXTI13 [2:0]			Res.	EXTI12 [2:0]		
	Reset value																		0	0	0		0	0	0		0	0	0		0	0	0
0x018	SYSCFG_SCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKASRAMBSY	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBSY	SRAM2ER
	Reset value																								0							0	0
0x01C	SYSCFG_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPF	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL
	Reset value																								0					0	0	0	0
0x020	SYSCFG_SWPR	P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP	P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SYSCFG_SKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x028 to 0x204	Reserved	Reserved																															
0x208	SYSCFG_RFDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFTBSEL
	Reset value																																0

Refer to *Section 2.4* for the register boundary addresses.

10 Peripherals interconnect matrix

10.1 Introduction

Several peripherals have direct connections between them, enabling autonomous communication and/or synchronization between them. This saves CPU resources and, consequently power consumption. In addition, these hardware connections remove software latency and result in more predictable system design.

Depending on peripherals, these interconnections can operate in Run, Sleep, LPRun, LPSleep, Stop 0, Stop 1 and Stop 2 modes.

10.2 Connection summary

Table 65. STM32WLEx peripherals interconnect matrix(1) (2)

Source	Destination													
	TIM1	TIM2	TIM16	TIM17	LPTIM1	LPTIM2	LPTIM3	ADC	DAC	COMP1	COMP2	DMAMUX1	IRTIM	SUBGHZSPI
TIM1	-	*1*	-	-	-	-	-	*3*	*3*	*8*	*8*	-	-	-
TIM2	*1*	-	-	-	-	-	-	*3*	*3*	*8*	*8*	-	-	-
TIM16	-	-	-	-	-	-	-	-	-	-	-	-	*12*	-
TIM17	*1*	-	-	-	-	-	-	-	-	-	-	-	*12*	-
LPTIM1	-	-	-	-	-	-	*2*	-	*4*	-	-	*13*	-	-
LPTIM2	-	-	-	-	-	-	*2*	-	*4*	-	-	*13*	-	-
LPTIM3	-	-	-	-	-	-	-	-	-	-	-	*13*	-	*14*
ADC	*5*	-	-	-	-	-		-	-	-	-	-	-	-
Temperature sensor	-	-	-	-	-	-	-	*9*	-	-	-	-	-	-
VBAT	-	-	-	-	-	-	-	*9*	-	-	-	-	-	-
VREFINT	-	-	-	-	-	-	-	*9*	-	-	-	-	-	-
HSE32	-	-	-	*6*	-	-	-	-	-	-	-	-	-	-
LSE	-	*6*	*6*	-	-	-	-	-	-	-	-	-	-	-
MSI	-	-	-	*6*	-	-	-	-	-	-	-	-	-	-
LSI	-	-	*6*	-	-	-	-	-	-	-	-	-	-	-
MCO	-	-	-	*6*	-	-	-	-	-	-	-	-	-	-
GPIO EXTI	-	-	-	-	-	-	-	*3*	*3*	-	-	*13*	-	-
RTC	-	-	*6*	-	*7*	*7*	-	-	-	-	-	-	-	-
TAMP	-	-	-	-	*7*	*7*	-	-	-	-	-	-	-	-

Table 65. STM32WLEx peripherals interconnect matrix(1) (2) (continued)

Source	Destination													
	TIM1	TIM2	TIM16	TIM17	LPTIM1	LPTIM2	LPTIM3	ADC	DAC	COMP1	COMP2	DMAMUX1	IRTIM	SUBGHZSPI
COMP1	*10*	*10*	*10*	*10*	*7*	*7*	-	-	-	-	-	-	-	-
COMP2	*10*	*10*	*10*	*10*	*7*	*7*	-	-	-	-	-	-	-	-
SYST ERR	*11*	-	*11*	*11*	-	-	-	-	-	-	-	-	-	-

1. Numbers in this table are links to corresponding subsections of *Section 10.3: Interconnection details*.
2. The "-" symbol in grayed cells means no interconnect.

10.3 Interconnection details

10.3.1 From timer (TIM1/TIM2/TIM17) to timer (TIM1/TIM2)

Purpose

Some timers are linked together internally for synchronization or chaining.

When one timer is configured in Master mode, it can reset, start, stop or clock the counter of another timer configured in Slave mode. A description of the feature is provided in *Section 23.3.26: Timer synchronization*.

The synchronization modes are detailed in the following sections:

- *Section 23.3.26: Timer synchronization* for advanced-control timers (TIM1)
- *Section 24.3.18: Timers and external trigger synchronization* for general-purpose timers (TIM2)

Triggering signals

The output (from master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1) following a configurable timer event. The input (to slave) is on signals TIMx_ITR0/ITR1/ITR2/ITR3.

The input and output signals for TIM1 are shown in *Figure 114: Advanced-control timer block diagram*.

The possible master/slave connections are given in tables below:

- *Table 167: TIM1 internal trigger connection* for TIM1
- *Table 171: TIM2 internal trigger connection* for TIM2

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.2 From timer (LPTIM1/LPTIM2) to timer (LPTIM3)

Purpose

Some timers are linked together internally for synchronization or chaining.

When one timer is configured in Master mode, it can reset, start, stop or clock the counter of another timer configured in Slave mode. A description of the feature is provided in *Section 23.3.26: Timer synchronization*.

Triggering signals

The output is on signals LPTIMx_OUT following a configurable timer event. The input (to slave) is on signals LPTIM3_ETR.

The input and output signals for LPTIM are shown in *Figure 248: Low-power timer block diagram*.

The possible connections are given in *Table 182: LPTIM3 external trigger connection*.

Active power modes

Run, Sleep, LPRun, LPSleep, Stop 0, Stop 1

10.3.3 From timer (TIM1/TIM2) and GPIO pin EXTI to ADC/DAC

Purpose

Advanced-control timer TIM1, general-purpose timer TIM2 and GPIO pin EXTI can be used to generate an ADC/DAC trigger event.

TIMx synchronization is described in *Section 23.3.27: ADC synchronization*.

GPIO pin EXTI mux is described in *Section 9: System configuration controller (SYSCFG)*.

ADC synchronization is described in *Section 16.4: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)*.

DAC synchronization is described in *Section 17.4.7: DAC trigger selection*.

Triggering signals

The output from timer is on signals TIMx_TRGO, TIMx_TRGO2, TIMx_CCx, TIMx_CHn event. The output from GPIO pin is on EXTI mux signal from SYSCFG.

The input to ADC is on signals TRG[7:0].

The connection between timers, GPIO pin EXTI mux and ADC, is provided in *Table 90: External triggers*.

The input to DAC is on signals dac_ch1_trg[15:0].

The connection between timers, GPIO pin EXTI mux and DAC, is provided in *Table 102: DAC interconnection*.

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.4 From timer (LPTIM1/LPTIM2) to DAC

Purpose

Low-power timer LPTIM1/LPTIM2 can be used to generate an DAC trigger event.

DAC triggering is described in *Section 17.4.7: DAC trigger selection*.

Triggering signals

The output from low-power timer is on signals LPTIMx_OUT event.

The input to DAC is on signals dac_ch1_trg[15:0].

The connection between timers and DAC is provided in *Table 102: DAC interconnection*.

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.5 From ADC to timer (TIM1)

Purpose

ADC can provide trigger event through watchdog signals to advanced-control timers (TIM1).

A description of the ADC analog watchdog setting is provided in *Section 16.8.2: Analog watchdog*.

Trigger settings on the timer are provided in *Section 23.3.4: External trigger input*.

Triggering signals

The output (from ADC) is on signals ADC_AWDx_OUT, x = 1, 2, 3 (3 watchdogs on ADC) and the input (to timer) on signal TIMx_ETR (external trigger).

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.6 From HSE32, LSE, LSI, MSI, MCO, RTC to timers (TIM2/TIM16/TIM17)

Purpose

External clocks (HSE32, LSE), internal clocks (LSI, MSI), microcontroller output clock (MCO), GPIO and RTC wakeup interrupt can be used as input to general-purpose timers (TIM16/17) channel 1.

This makes possible calibration of the HSI16/MSI system clocks (with TIM16 and LSE) or of the LSI (with TIM16 and HSE32). This is also used to precisely measure LSI (with TIM16 and HSI16) or MSI (with TIM17 and HSI16) oscillator frequency.

When the low-speed external (LSE) oscillator is used, no additional hardware connections are required.

This feature is described in *Section 6.2.20: Internal/external clock measurement with TIM16/TIM17*.

External clock LSE can be used as input to general-purpose timers (TIM2) on TIM2_ETR pin (see *TIM2 option register 1 (TIM2_OR1)*).

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.7 From RTC, TAMP, COMP1, COMP2 to low-power timers (LPTIM1/LPTIM2)

Purpose

RTC alarm A/B, TAMP_IN1/2/3 input detection and COMP1/2_OUT can be used as trigger to start LPTIM1/LPTIM2 counters.

Triggering signals

This trigger feature is described in *Section 26.4.7: Trigger multiplexer* (and following sections).

The input selection is described in *Table 180: LPTIM1 external trigger connection* and *Table 181: LPTIM2 external trigger connection*.

Active power modes

Run, Sleep, LPRun, LPSleep, Stop 0, Stop 1, Stop 2 (LPTIM1 only)

10.3.8 From timer (TIM1/TIM2) to comparators (COMP1/COMP2)

Purpose

Advanced-control timer (TIM1) and general-purpose timer (TIM2) can be used as blanking window input to COMP1/COMP2.

The blanking function is described in *Section 19.3.7: Comparator output blanking function*.

The blanking sources are given in the following registers:

- *COMP1 control and status register (COMP1_CSR)* bits 20:18 BLANKING[2:0]
- *COMP2 control and status register (COMP2_CSR)* bits 20:18 BLANKING[2:0]

Triggering signals

Timer output signals TIMx_OCx are the inputs to blanking source of COMP1/COMP2.

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.9 From internal analog to ADC

Purpose

Internal temperature sensor (V_{TS}), Internal reference voltage (V_{REFINT}) and V_{BAT} monitoring channel are connected to ADC input channel.

This is according to the following sections:

- *Section 16.2: ADC main features*
- *Section 16.3.8: Channel selection (CHSEL, SCANDIR, CHSELRMOD)*
- *Section 16.9: Temperature sensor and internal reference voltage*
- *Section 16.10: Battery voltage monitoring*

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.10 From comparators (COMP1/COMP2) to timers (TIM1/TIM2/TIM16/TIM17)

Purpose

Comparators (COMP1/COMP2) output values can be connected to timers TIM1/TIM2/TIM16/TIM17 input captures or TIMx_ETR signals.

Comparators (COMP1/COMP2) output values can also generate break input signals for timer TIM1 on input pins TIMx_BKIN or TIMx_BKIN2 through GPIO alternate function selection using open drain connection of I/Os.

The possible connections are given in the following sections:

- *Section 23.4.23: TIM1 option register 1 (TIM1_OR1)*
- *Section 23.4.28: TIM1 Alternate function register 2 (TIM1_AF2)*
- *Section 24.4.22: TIM2 option register 1 (TIM2_OR1)*
- *Section 24.4.23: TIM2 alternate function option register 1 (TIM2_AF1)*
- *Section 25.4.21: TIM17 alternate function register 1 (TIM17_AF1)*

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.11 From system errors to timers (TIM1/TIM16/TIM17)

Purpose

CSS, CPU hard fault, RAM parity error, FLASH ECC double error detection and PVD can generate system errors in the form of timer break toward timers (TIM1/TIM16/TIM17).

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

List of possible source of break are described in the following sections:

- *Section 23.3.16: Using the break function* (TIM1)
- *Section 25.3.11: Using the break function* (TIM16/TIM17)

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.12 From timers (TIM16/TIM17) to IRTIM

Purpose

General-purpose timer (TIM16/TIM17) output channels TIMx_OC1 are used to generate the waveform of infrared signal output.

The functionality is described in *Section 27: Infrared interface (IRTIM)*.

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.13 From timer (LPTIM1/LPTIM2/LPTIM3/GPIO pin EXTI) to DMAMUX1 trigger

Purpose

Low-power timers LPTIM1/LPTIM2/LPTIM3 and GPIO pin EXTI mux in SYSCFG, can be used to generate a DMAMUX1 trigger event.

GPIO pin EXTI mux is described in *Section 9: System configuration controller (SYSCFG)*.

DMAMUX1 triggering is described in *Section 12.3.2: DMAMUX1 mapping*.

Triggering signals

The output from low-power timer is on signals LPTIMx_OUT event. The output from GPIO pin is on EXTI mux signal from SYSCFG.

The input to DMAMUX1 is on signals trigger input [20:0].

The connection between timers, GPIO pin EXTI mux and DMAMUX1, is provided in *Table 72: DMAMUX1: assignment of multiplexer inputs to resources*.

Active power modes

Run, Sleep, LPRun, LPSleep

10.3.14 From timer (LPTIM3) to sub-GHz radio SPI NSS

Purpose

Low-power timer LPTIM3 can be used to generate a sub-GHz radio SPI NSS event.

Triggering signals

The output from low-power timer is on signal LPTIM3_OUT event.

The connection between timers and sub-GHz radio SPI NSS is provided in *PWR sub-GHz SPI control register (PWR_SUBGHZSPICR)*.

Active power modes

Run, Sleep, LPRun, LPSleep

11 Direct memory access controller (DMA)

11.1 Introduction

The direct memory access (DMA) controller is a bus master and system peripheral.

The DMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.

The DMA controller features a single AHB master architecture.

There are two instances of DMA, DMA1 and DMA2.

Each channel is dedicated to managing memory access requests from one or more peripherals. Each DMA includes an arbiter for handling the priority between DMA requests.

11.2 DMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access, as source and destination, to on-chip memory-mapped devices such as Flash memory, SRAM, and AHB and APB peripherals
- All DMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (4 levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to 2^{18} - 1
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.
- Privileged/unprivileged support:
  - Support for AHB privileged and unprivileged DMA transfers, independently of a channel level
  - Privileged-aware AHB slave port

11.3 DMA implementation

11.3.1 DMA1 and DMA2

DMA1 and DMA2 are implemented with the hardware configuration parameters shown in *Table 66*.

Table 66. DMA1 and DMA2 implementation

Feature	DMA1	DMA2
Number of channels	7	7

11.3.2 DMA request mapping

The DMA controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX peripheral.

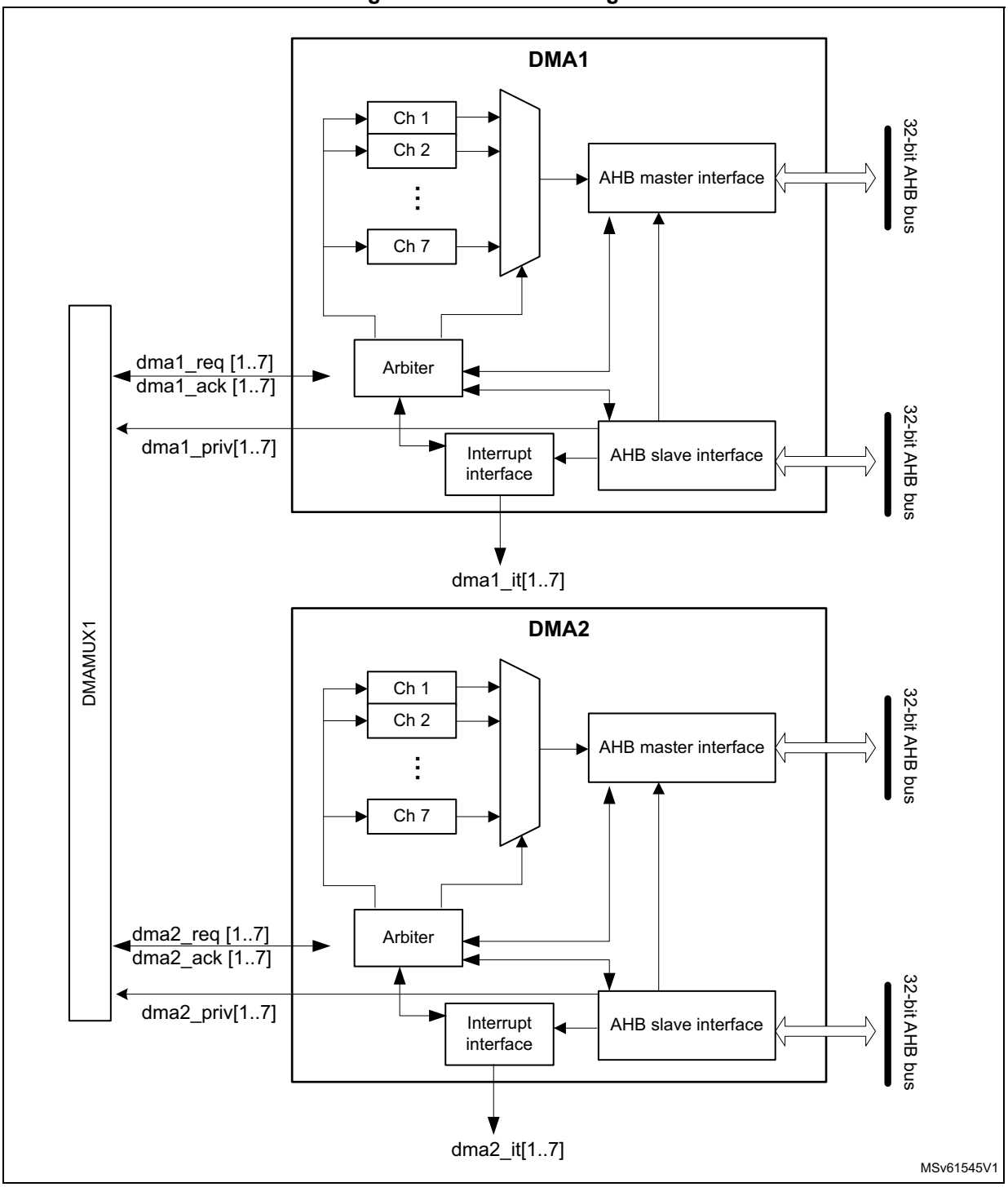
For the mapping of the different requests, refer to the *Section 12.3: DMAMUX implementation*.

11.4 DMA functional description

11.4.1 DMA block diagram

The DMA block diagram is shown in *Figure 36*.

Figure 36. DMA block diagram



The DMA controller performs direct memory transfer by sharing the AHB system bus with other system masters. The bus matrix implements round-robin scheduling. DMA requests may stop the CPU access to the system bus for a number of bus cycles, when CPU and DMA target the same destination (memory or peripheral).

According to its configuration through the AHB slave interface, the DMA controller arbitrates between the DMA channels and their associated received requests. The DMA controller also schedules the DMA data transfers over the single AHB port master.

The DMA controller generates a privileged bus, to keep the DMAMUX peripheral informed of the privileged or unprivileged state of each channel x.

The DMA controller generates an interrupt per channel to the interrupt controller.

11.4.2 DMA pins and internal signals

Table 67. DMA internal input/output signals

Signal name	Signal type	Description
dma_req[x]	Input	DMA channel x request
dma_ack[x]	Output	DMA channel x acknowledge
dma_it[x]	Output	DMA channel x interrupt
dma_priv[x]	Output	DMA channel x privileged state

11.4.3 DMA transfers

The software configures the DMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB privileged or unprivileged bus transfers.

A DMA block transfer may be requested from a peripheral, or triggered by the software in case of memory-to-memory transfer.

After an event, the following steps of a single DMA transfer occur:

1. The peripheral sends a single DMA request signal to the DMA controller.
2. The DMA controller serves the request, depending on the priority of the channel associated to this peripheral request.
3. As soon as the DMA controller grants the peripheral, an acknowledge is sent to the peripheral by the DMA controller.
4. The peripheral releases its request as soon as it gets the acknowledge from the DMA controller.
5. Once the request is de-asserted by the peripheral, the DMA controller releases the acknowledge.

The peripheral may order a further single request and initiate another single DMA transfer.

The request/acknowledge protocol is used when a peripheral is either the source or the destination of the transfer. For example, in case of memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the DMA controller. The DMA controller reads then a single data in the memory and writes this data to the peripheral.

For a given channel x, a DMA block transfer consists of a repeated sequence of:

- a single DMA transfer, encapsulating two AHB transfers of a single data, over the DMA AHB bus master:
  - a single data read (byte, half-word or word) from the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
    The start address used for the first single transfer is the base address of the peripheral or memory, and is programmed in the DMA_CPARx or DMA_CMARx register.
  - a single data write (byte, half-word or word) to the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
    The start address used for the first transfer is the base address of the peripheral or memory, and is programmed in the DMA_CPARx or DMA_CMARx register.
- post-decrementing of the programmed DMA_CNDTRx register
  This register contains the remaining number of data items to transfer (number of AHB 'read followed by write' transfers).

This sequence is repeated until DMA_CNDTRx is null.

Note: The AHB master bus source/destination address must be aligned with the programmed size of the transferred single data to the source/destination.

11.4.4 DMA arbitration

The DMA arbiter manages the priority between the different channels.

When an active channel x is granted by the arbiter (hardware requested or software triggered), a single DMA transfer is issued (such as a AHB 'read followed by write' transfer of a single data). Then, the arbiter considers again the set of active channels and selects the one with the highest priority.

The priorities are managed in two stages:

- software: priority of each channel is configured in the DMA_CCRx register, to one of the four different levels:
  - very high
  - high
  - medium
  - low
- hardware: if two requests have the same software priority level, the channel with the lowest index gets priority. For example, channel 2 gets priority over channel 4.

When a channel x is programmed for a block transfer in memory-to-memory mode, re arbitration is considered between each single DMA transfer of this channel x. Whenever there is another concurrent active requested channel, the DMA arbiter automatically alternates and grants the other highest-priority requested channel, which may be of lower priority than the memory-to-memory channel.

11.4.5 DMA channels

Each channel may handle a DMA transfer between a peripheral register located at a fixed address, and a memory address. The amount of data items to transfer is programmable.

The register that contains the amount of data items to transfer is decremented after each transfer.

A DMA channel is programmed at block transfer level.

Programmable data sizes

The transfer sizes of a single data (byte, half-word, or word) to the peripheral and memory are programmable through, respectively, the PSIZE[1:0] and MSIZE[1:0] fields of the DMA_CCRx register.

Pointer incrementation

The peripheral and memory pointers may be automatically incremented after each transfer, depending on the PINC and MINC bits of the DMA_CCRx register.

If the **incremented mode** is enabled (PINC or MINC set to 1), the address of the next transfer is the address of the previous one incremented by 1, 2 or 4, depending on the data size defined in PSIZE[1:0] or MSIZE[1:0]. The first transfer address is the one programmed in the DMA_CPARx or DMA_CMARx register. During transfers, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel x is configured in **non-circular mode**, no DMA request is served after the last data transfer (once the number of single data to transfer reaches zero). The DMA channel must be disabled in order to reload a new number of data items into the DMA_CNDTRx register.

Note: If the channel x is disabled, the DMA registers are not reset. The DMA channel registers (DMA_CCRx, DMA_CPARx and DMA_CMARx) retain the initial values programmed during the channel configuration phase.

In **circular mode**, after the last data transfer, the DMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the DMA_CPARx and DMA_CMARx registers.

Privileged / unprivileged mode

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged software via the PRIV bit of the DMA_CCRx register.

When a channel x is configured in privileged mode, the following access controls rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for the privileged state of this channel x (PRIV bit of the DMA_CCRx register) which is readable by an unprivileged software.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) mode, the AHB master transfers from the source and to the destination, are privileged (respectively unprivileged).

DMA generates a privileged bus, dma_priv[7:0], reflecting the PRIV bit of the DMA_CCRx register, in order to keep the other hardware peripherals, like DMAMUX, informed of the privileged / unprivileged state of each DMA channel x.

Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

1. Set a channel x to privileged or unprivileged, by a privileged write access to the privileged PRIV bit of the DMA_CCRx register.
2. Set the memory address in the DMA_CMARx register.
   The data is written to/read from the memory after the peripheral event or after the channel is enabled in memory-to-memory mode.
3. Configure the total number of data to transfer in the DMA_CNDTRx register.
   After each data transfer, this value is decremented.
4. Configure the parameters listed below in the DMA_CCRx register:
   – the channel priority
   – the data transfer direction
   – the circular mode
   – the peripheral and memory incremented mode
   – the peripheral and memory data size
   – the interrupt enable at half and/or full transfer and/or transfer error
5. Activate the channel by setting the EN bit in the DMA_CCRx register.

A channel, as soon as enabled, may serve any DMA request from the peripheral connected to this channel, or may start a memory-to-memory block transfer.

Note: The two last steps of the channel configuration procedure may be merged into a single access to the DMA_CCRx register, to configure and enable the channel.

Channel state and disabling a channel

A channel x in active state is an enabled channel (read DMA_CCRx.EN = 1). An active channel x is a channel that must have been enabled by the software (DMA_CCRx.EN set to 1) and afterwards with no occurred transfer error (DMA_ISR.TEIFx = 0). In case there is a transfer error, the channel is automatically disabled by hardware (DMA_CCRx.EN = 0).

The three following use cases may happen:

- Suspend and resume a channel

  This corresponds to the two following actions:
  – An active channel is disabled by software (writing DMA_CCRx.EN = 0 whereas DMA_CCRx.EN = 1).
  – The software enables the channel again (DMA_CCRx.EN set to 1) without reconfiguring the other channel registers (such as DMA_CNDTRx, DMA_CPARx and DMA_CMARx).

  This case is not supported by the DMA hardware, that does not guarantee that the remaining data transfers are performed correctly.
- Stop and abort a channel

  If the application does not need any more the channel, this active channel can be disabled by software. The channel is stopped and aborted but the DMA_CNDTRx

register content may not correctly reflect the remaining data transfers versus the aborted source and destination buffer/register.

- Abort and restart a channel

  This corresponds to the software sequence: disable an active channel, then reconfigure the channel and enable it again.

  This is supported by the hardware if the following conditions are met:

  – The application guarantees that, when the software is disabling the channel, a DMA data transfer is not occurring at the same time over its master port. For example, the application can first disable the peripheral in DMA mode, in order to ensure that there is no pending hardware DMA request from this peripheral.

  – The software must operate separated write accesses to the same DMA_CCRx register: First disable the channel. Second reconfigure the channel for a next block transfer including the DMA_CCRx if a configuration change is needed. There are read-only DMA_CCRx register fields when DMA_CCRx.EN=1. Finally enable again the channel.

When a channel transfer error occurs, the EN bit of the DMA_CCRx register is cleared by hardware. This EN bit can not be set again by software to re-activate the channel x, until the TEIFx bit of the DMA_ISR register is set.

Circular mode (in memory-to-peripheral/peripheral-to-memory transfers)

The circular mode is available to handle circular buffers and continuous data flows (such as ADC scan mode). This feature is enabled using the CIRC bit in the DMA_CCRx register.

Note: The circular mode must not be used in memory-to-memory mode. Before enabling a channel in circular mode (CIRC = 1), the software must clear the MEM2MEM bit of the DMA_CCRx register. When the circular mode is activated, the amount of data to transfer is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

In order to stop a circular transfer, the software needs to stop the peripheral from generating DMA requests (such as quit the ADC scan mode), before disabling the DMA channel. The software must explicitly program the DMA_CNDTRx value before starting/enabling a transfer, and after having stopped a circular transfer.

Memory-to-memory mode

The DMA channels may operate without being triggered by a request from a peripheral. This mode is called memory-to-memory mode, and is initiated by software.

If the MEM2MEM bit in the DMA_CCRx register is set, the channel, if enabled, initiates transfers. The transfer stops once the DMA_CNDTRx register reaches zero.

Note: The memory-to-memory mode must not be used in circular mode. Before enabling a channel in memory-to-memory mode (MEM2MEM = 1), the software must clear the CIRC bit of the DMA_CCRx register.

Peripheral-to-peripheral mode

Any DMA channel can operate in peripheral-to-peripheral mode:

- when the hardware request from a peripheral is selected to trigger the DMA channel

  This peripheral is the DMA initiator and paces the data transfer from/to this peripheral to/from a register belonging to another memory-mapped peripheral (this one being not configured in DMA mode).
- when no peripheral request is selected and connected to the DMA channel

  The software configures a register-to-register transfer by setting the MEM2MEM bit of the DMA_CCRx register.

Programming transfer direction, assigning source/destination

The value of the DIR bit of the DMA_CCRx register sets the direction of the transfer, and consequently, it identifies the source and the destination, regardless the source/destination type (peripheral or memory):

- **DIR = 1** defines typically a memory-to-peripheral transfer. More generally, if DIR = 1:
  - The **source** attributes are defined by the DMA_MARx register, the MSIZE[1:0] field and MINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the source peripheral in peripheral-to-peripheral mode.
  - The **destination** attributes are defined by the DMA_PARx register, the PSIZE[1:0] field and PINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the destination memory in memory-to-memory mode.
- **DIR = 0** defines typically a peripheral-to-memory transfer. More generally, if DIR = 0:
  - The **source** attributes are defined by the DMA_PARx register, the PSIZE[1:0] field and PINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the source memory in memory-to-memory mode
  - The **destination** attributes are defined by the DMA_MARx register, the MSIZE[1:0] field and MINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the destination peripheral in peripheral-to-peripheral mode.

11.4.6 DMA data width, alignment and endianness

When PSIZE[1:0] and MSIZE[1:0] are not equal, the DMA controller performs some data alignments as described in *Table 68*.

Table 68. Programmable data width and endian behavior (when PINC = MINC = 1)

Source port width (MSIZE if DIR = 1, else PSIZE)	Destination port width (PSIZE if DIR = 1, else MSIZE)	Number of data items to transfer (NDT)	Source content: address / data (DMA_CMARx if DIR = 1, else DMA_CPARx)	DMA transfers	Destination content: address / data (DMA_CPARx if DIR = 1, else DMA_CMARx)
8	8	8	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write B0[7:0] @0x0 2: read B1[7:0] @0x1 then write B1[7:0] @0x1 3: read B2[7:0] @0x2 then write B2[7:0] @0x2 4: read B3[7:0] @0x3 then write B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 00B0[15:0] @0x0 2: read B1[7:0] @0x1 then write 00B1[15:0] @0x2 3: read B2[7:0] @0x2 then write 00B2[15:0] @0x4 4: read B3[7:0] @0x3 then write 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 000000B0[31:0] @0x0 2: read B1[7:0] @0x1 then write 000000B1[31:0] @0x4 3: read B2[7:0] @0x2 then write 000000B2[31:0] @0x8 4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0 2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1 3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2 4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B1B0[15:0] @0x0 2: read B3B2[15:0] @0x2 then write B3B2[15:0] @0x2 3: read B5B4[15:0] @0x4 then write B5B4[15:0] @0x4 4: read B7B6[15:0] @0x6 then write B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0 2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4 3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8 4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x1 3: read BBBAB9B8[31:0] @0x8 then write B8[7:0] @0x2 4: read BFBEBDBC[31:0] @0xC then write BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B1B0[15:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B5B4[15:0] @0x2 3: read BBBAB9B8[31:0] @0x8 then write B9B8[15:0] @0x4 4: read BFBEBDBC[31:0] @0xC then write BDBC[15:0] @0x6	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32	32	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B3B2B1B0[31:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B7B6B5B4[31:0] @0x4 3: read BBBAB9B8[31:0] @0x8 then write BBBAB9B8[31:0] @0x8 4: read BFBEBDBC[31:0] @0xC then write BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

Addressing AHB peripherals not supporting byte/half-word write transfers

When the DMA controller initiates an AHB byte or half-word write transfer, the data are duplicated on the unused lanes of the AHB master 32-bit data bus (HWDATA[31:0]).

When the AHB slave peripheral does not support byte or half-word write transfers and does not generate any error, the DMA controller writes the 32 HWDATA bits as shown in the two examples below:

- To write the half-word 0xABCD, the DMA controller sets the HWDATA bus to 0xABCDABCD with a half-word data size (HSIZE = HalfWord in AHB master bus).
- To write the byte 0xAB, the DMA controller sets the HWDATA bus to 0xABABABAB with a byte data size (HSIZE = Byte in the AHB master bus).

Assuming the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take into account the HSIZE data, any AHB byte or half-word transfer is changed into a 32-bit APB transfer as described below:

- An AHB byte write transfer of 0xB0 to one of the 0x0, 0x1, 0x2 or 0x3 addresses, is converted to an APB word write transfer of 0xB0B0B0B0 to the 0x0 address.
- An AHB half-word write transfer of 0xB1B0 to the 0x0 or 0x2 addresses, is converted to an APB word write transfer of 0xB1B0B1B0 to the 0x0 address.

11.4.7 DMA error management

A DMA transfer error is generated when reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or write access, the faulty channel x is automatically disabled through a hardware clear of its EN bit in the corresponding DMA_CCRx register.

The TEIFx bit of the DMA_ISR register is set. An interrupt is then generated if the TEIE bit of the DMA_CCRx register is set.

The EN bit of the DMA_CCRx register can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

When the software is notified with a transfer error over a channel which involves a peripheral, the software has first to stop this peripheral in DMA mode, in order to disable any pending or future DMA request. Then software may normally reconfigure both DMA and the peripheral in DMA mode for a new transfer.

11.5 DMA interrupts

An interrupt can be generated on a half transfer, transfer complete or transfer error for each DMA channel x. Separate interrupt enable bits are available for flexibility.

Table 69. DMA interrupt requests

Interrupt request	Interrupt event	Event flag	Interrupt enable bit
Channel x interrupt	Half transfer on channel x	HTIFx	HTIEx
	Transfer complete on channel x	TCIFx	TCIEx
	Transfer error on channel x	TEIFx	TEIEx
	Half transfer or transfer complete or transfer error on channel x	GIFx	-

11.6 DMA registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The DMA registers have to be accessed by words (32-bit).

11.6.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

This register may mix privileged and unprivileged information, depending on the privileged mode of each channel (PRIV bit of the DMA_CCRx register). A privileged software can read the full interrupt status. An unprivileged software is restricted to read the status of unprivileged channel(s), other privileged bit fields returning zero.

Every status bit is cleared by hardware when the software sets the corresponding clear bit or the corresponding global clear bit CGIFx, in the DMA_IFCR register, provided that, if the channel x is in privileged mode, then the software access to DMA_IFCR is also privileged.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TEIF7**: transfer error (TE) flag for channel 7
0: no TE event
1: a TE event occurred

Bit 26 **HTIF7**: half transfer (HT) flag for channel 7
0: no HT event
1: a HT event occurred

Bit 25 **TCIF7**: transfer complete (TC) flag for channel 7
0: no TC event
1: a TC event occurred

Bit 24 **GIF7**: global interrupt flag for channel 7
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 23 **TEIF6**: transfer error (TE) flag for channel 6
0: no TE event
1: a TE event occurred

Bit 22 **HTIF6**: half transfer (HT) flag for channel 6
0: no HT event
1: a HT event occurred

Bit 21 **TCIF6**: transfer complete (TC) flag for channel 6
0: no TC event
1: a TC event occurred

Bit 20 **GIF6**: global interrupt flag for channel 6
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 19 **TEIF5**: transfer error (TE) flag for channel 5
0: no TE event
1: a TE event occurred

Bit 18 **HTIF5**: half transfer (HT) flag for channel 5
0: no HT event
1: a HT event occurred

Bit 17 **TCIF5**: transfer complete (TC) flag for channel 5
0: no TC event
1: a TC event occurred

Bit 16 **GIF5**: global interrupt flag for channel 5
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 15 **TEIF4**: transfer error (TE) flag for channel 4
0: no TE event
1: a TE event occurred

Bit 14 **HTIF4**: half transfer (HT) flag for channel 4
0: no HT event
1: a HT event occurred

Bit 13 **TCIF4**: transfer complete (TC) flag for channel 4
0: no TC event
1: a TC event occurred

Bit 12 **GIF4**: global interrupt flag for channel 4
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 11 **TEIF3**: transfer error (TE) flag for channel 3
0: no TE event
1: a TE event occurred

Bit 10 **HTIF3**: half transfer (HT) flag for channel 3
0: no HT event
1: a HT event occurred

Bit 9 **TCIF3**: transfer complete (TC) flag for channel 3
0: no TC event
1: a TC event occurred

Bit 8 **GIF3**: global interrupt flag for channel 3
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 7 **TEIF2**: transfer error (TE) flag for channel 2
0: no TE event
1: a TE event occurred

Bit 6 **HTIF2**: half transfer (HT) flag for channel 2
0: no HT event
1: a HT event occurred

Bit 5 **TCIF2**: transfer complete (TC) flag for channel 2
0: no TC event
1: a TC event occurred

Bit 4 **GIF2**: global interrupt flag for channel 2
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 3 **TEIF1**: transfer error (TE) flag for channel 1
0: no TE event
1: a TE event occurred

Bit 2 **HTIF1**: half transfer (HT) flag for channel 1
0: no HT event
1: a HT event occurred

Bit 1 **TCIF1**: transfer complete (TC) flag for channel 1
0: no TC event
1: a TC event occurred

Bit 0 **GIF1**: global interrupt flag for channel 1
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

11.6.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

This register may mix privileged and unprivileged information, depending on the privileged mode of each channel (PRIV bit of the DMA_CCRx register).

A privileged software is able to set any flag clear bit of the DMA_IFCR, and order DMA hardware to clear any corresponding flag(s) in the DMA_ISR register.

An unprivileged software is restricted to order DMA hardware to clear the unprivileged flag(s) in the DMA_ISR, by setting any unprivileged corresponding flag clear bit(s) of the DMA_IFCR register.

Setting the global clear bit CGIFx of the channel x in this DMA_IFCR register, causes the DMA hardware to clear the corresponding GIFx bit and any individual flag among TEIFx, HTIFx, TCIFx, in the DMA_ISR register.

Setting any individual clear bit among CTEIFx, CHTIFx, CTCIFx in this DMA_IFCR register, causes the DMA hardware to clear the corresponding individual flag and the global flag GIFx in the DMA_ISR register, provided that none of the two other individual flags is set.

Writing 0 into any flag clear bit has no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **CTEIF7**: transfer error flag clear for channel 7

Bit 26 **CHTIF7**: half transfer flag clear for channel 7

Bit 25 **CTCIF7**: transfer complete flag clear for channel 7

Bit 24 **CGIF7**: global interrupt flag clear for channel 7

Bit 23 **CTEIF6**: transfer error flag clear for channel 6

Bit 22 **CHTIF6**: half transfer flag clear for channel 6

Bit 21 **CTCIF6**: transfer complete flag clear for channel 6

Bit 20 **CGIF6**: global interrupt flag clear for channel 6

Bit 19 **CTEIF5**: transfer error flag clear for channel 5

Bit 18 **CHTIF5**: half transfer flag clear for channel 5

Bit 17 **CTCIF5**: transfer complete flag clear for channel 5

Bit 16 **CGIF5**: global interrupt flag clear for channel 5

Bit 15 **CTEIF4**: transfer error flag clear for channel 4

Bit 14 **CHTIF4**: half transfer flag clear for channel 4

Bit 13 **CTCIF4**: transfer complete flag clear for channel 4

Bit 12 **CGIF4**: global interrupt flag clear for channel 4

Bit 11 **CTEIF3**: transfer error flag clear for channel 3

Bit 10 **CHTIF3**: half transfer flag clear for channel 3

Bit 9 **CTCIF3**: transfer complete flag clear for channel 3

Bit 8 **CGIF3**: global interrupt flag clear for channel 3

Bit 7 **CTEIF2**: transfer error flag clear for channel 2

Bit 6 **CHTIF2**: half transfer flag clear for channel 2

Bit 5 **CTCIF2**: transfer complete flag clear for channel 2

Bit 4 **CGIF2**: global interrupt flag clear for channel 2

Bit 3 **CTEIF1**: transfer error flag clear for channel 1

Bit 2 **CHTIF1**: half transfer flag clear for channel 1

Bit 1 **CTCIF1**: transfer complete flag clear for channel 1

Bit 0 **CGIF1**: global interrupt flag clear for channel 1

11.6.3 DMA channel x configuration register (DMA_CCRx)

Address offset: 0x08 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

This register contains privileged information: the privileged state of the channel x (PRIV control bit).

Modifying the PRIV bit must be performed by a privileged write access to this register. Except PRIV control bit, any other register field is non-readable by an unprivileged software if the PRIV bit is set.

The register fields/bits PRIV, MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.	Res.	Res.	Res.
											rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **PRIV**: privileged mode

This bit can only be set and cleared by a privileged software.

0: disabled

1: enabled

This bit must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **MEM2MEM**: memory-to-memory mode

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 13:12 **PL[1:0]**: priority level

00: low

01: medium

10: high

11: very high

Note: This field is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 11:10 **MSIZE[1:0]**: memory size

Defines the data size of each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

Note: This field is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 9:8 **PSIZE[1:0]**: peripheral size

Defines the data size of each DMA transfer to the identified peripheral.

In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

Note: This field is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 7 **MINC**: memory increment mode

Defines the increment mode for each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 6 **PINC**: peripheral increment mode

Defines the increment mode for each DMA transfer to the identified peripheral.

n memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 5 **CIRC**: circular mode

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 4 **DIR**: data transfer direction

This bit must be set only in memory-to-peripheral and peripheral-to-memory modes.

0: read from peripheral

- Source attributes are defined by PSIZE and PINC, plus the DMA_CPARx register. This is still valid in a memory-to-memory mode.
- Destination attributes are defined by MSIZE and MINC, plus the DMA_CMARx register. This is still valid in a peripheral-to-peripheral mode.

1: read from memory

- Destination attributes are defined by PSIZE and PINC, plus the DMA_CPARx register. This is still valid in a memory-to-memory mode.
- Source attributes are defined by MSIZE and MINC, plus the DMA_CMARx register. This is still valid in a peripheral-to-peripheral mode.

Note: This bit is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 3 **TEIE**: transfer error interrupt enable

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 2 **HTIE**: half transfer interrupt enable

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 1 **TCIE**: transfer complete interrupt enable

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 0 **EN**: channel enable

When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged software if the channel is in privileged mode)

11.6.4 DMA channel x number of data to transfer register (DMA_CNDTRx)

Address offset: 0x0C + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:16]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **NDT[17:0]**: number of data to transfer (0 to 2^{18} - 1)

This field is updated by hardware when the channel is enabled:

- It is decremented after each single DMA 'read followed by write' transfer, indicating the remaining amount of data items to transfer.
- It is kept at zero when the programmed amount of data to transfer is reached, if the channel is not in circular mode (CIRC = 0 in the DMA_CCRx register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this field is zero, no transfer can be served whatever the channel status (enabled or not).

Note: This field is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

11.6.5 DMA channel x peripheral address register (DMA_CPARx)

Address offset: 0x10 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PA[31:0]**: peripheral address

It contains the base address of the peripheral data register from/to which the data is read/written.

When PSIZE[1:0] = 01 (16 bits), bit 0 of PA[31:0] is ignored. Access is automatically aligned to a half-word address.

When PSIZE = 10 (32 bits), bits 1 and 0 of PA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory destination address if DIR = 1 and the memory source address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral destination address DIR = 1 and the peripheral source address if DIR = 0.

Note: This register is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

11.6.6 DMA channel x memory address register (DMA_CMARx)

Address offset: 0x14 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MA[31:0]**: peripheral address

It contains the base address of the memory from/to which the data is read/written.

When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.

When MSIZE = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral source address DIR = 1 and the peripheral destination address if DIR = 0.

Note: This register is set and cleared by software (privileged software if the channel is in privileged mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

11.6.7 DMA register map

Table 70. DMA register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMA_ISR	Res.	Res.	Res.	Res.	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 70. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	DMA_IFCR	Res.	Res.	Res.	Res.	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value												0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	DMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved	Reserved.																															
0x01C	DMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value												0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CMAR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved	Reserved.																															
0x030	DMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value												0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_CMAR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	Reserved	Reserved.																															
0x044	DMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value												0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	DMA_CPAR4	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	DMA_CMAR4	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved	Reserved.																															

Table 70. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	DMA_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value												0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	DMA_CNDTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DMA_CPAR5	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	DMA_CMAR5	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved	Reserved.																															
0x06C	DMA_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value												0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	DMA_CNDTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	DMA_CPAR6	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	DMA_CMAR6	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved	Reserved.																															
0x080	DMA_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value												0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DMA_CNDTR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	DMA_CPAR7	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	DMA_CMAR7	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

12 DMA request multiplexer (DMAMUX)

12.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until it is served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is deasserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals and the DMA controllers of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in *Section 12.3.1*.

The assignment of DMAMUX request multiplexer inputs to the DMA request lines from peripherals and to the DMAMUX request generator outputs, the assignment of DMAMUX request multiplexer outputs to DMA controller channels, and the assignment of DMAMUX synchronizations and trigger inputs to internal and external signals depend on the product implementation, and are detailed in *Section 12.3.2*.

12.2 DMAMUX main features

- 14-channel programmable DMA request line multiplexer output
- 4-channel DMA request generator
- 21 trigger inputs to DMA request generator
- 21 synchronization inputs
- Per DMA request generator channel:
  - DMA request trigger input selector
  - DMA request counter
  - Event overrun flag for selected DMA request trigger input
- Per DMA request line multiplexer channel output:
  - 38 input DMA request lines from peripherals
  - One DMA request line output
  - Synchronization input selector
  - DMA request counter
  - Event overrun flag for selected synchronization input
  - One event output, for DMA request chaining
- Privileged / Unprivileged support:
  - Support for AHB privileged and unprivileged DMA transfers, independently, at a channel level.
  - Privileged-aware AHB slave port.

12.3 DMAMUX implementation

12.3.1 DMAMUX1 instantiation

DMAMUX1 instantiated with the hardware configuration parameters listed in the following table.

Table 71. DMAMUX instantiation

Feature	DMAMUX1
Number of DMAMUX output request channels	14
Number of DMAMUX request generator channels	4
Number of DMAMUX request trigger inputs	21
Number of DMAMUX synchronization inputs	21
Number of DMAMUX peripheral request inputs	38

12.3.2 DMAMUX1 mapping

The mapping of resources to DMAMUX1 is hardwired.

DMAMUX1 is used with DMA1 and DMA2

- DMAMUX1 channels 0 to 6 are connected to DMA1 channels 1 to 7
- DMAMUX1 channels 7 to 13 are connected to DMA2 channels 1 to 7

Table 72. DMAMUX1: assignment of multiplexer inputs to resources

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux_req_gen0	22	LPUART1_TX	43	Reserved
2	dmamux_req_gen1	23	TIM1_CH1	44	Reserved
3	dmamux_req_gen2	24	TIM1_CH2	45	Reserved
4	dmamux_req_gen3	25	TIM1_CH3	46	Reserved
5	ADC	26	TIM1_CH4	47	Reserved
6	DAC_OUT1	27	TIM1_UP	48	Reserved
7	SPI1_RX	28	TIM1_TRIG	49	Reserved
8	SPI1_TX	29	TIM1_COM	50	Reserved
9	SPI2_RX	30	TIM2_CH1	51	Reserved
10	SPI2_TX	31	TIM2_CH2	52	Reserved
11	I2C1_RX	32	TIM2_CH3	53	Reserved
12	I2C1_TX	33	TIM2_CH4	54	Reserved
13	I2C2_RX	34	TIM2_UP	55	Reserved
14	I2C2_TX	35	TIM16_CH1	56	Reserved
15	I2C3_RX	36	TIM16_UP	57	Reserved
16	I2C3_TX	37	TIM17_CH1	58	Reserved
17	USART1_RX	38	TIM17_UP	59	Reserved
18	USART1_TX	39	AES_IN	60	Reserved
19	USART2_RX	40	AES_OUT	61	Reserved
20	USART2_TX	41	SUBGHZSPI_RX	62	Reserved
21	LPUART1_RX	42	SUBGHZSPI_TX	63	Reserved

Table 73. DMAMUX1: assignment of trigger inputs to resources

Trigger input	Resource	Trigger input	Resource
0	EXTI LINE0	16	dmamux_evt0
1	EXTI LINE1	17	dmamux_evt1
2	EXTI LINE2	18	LPTIM1_OUT
3	EXTI LINE3	19	LPTIM2_OUT
4	EXTI LINE4	20	LPTIM3_OUT
5	EXTI LINE5	21	Reserved
6	EXTI LINE6	22	Reserved
7	EXTI LINE7	23	Reserved
8	EXTI LINE8	24	Reserved
9	EXTI LINE9	25	Reserved
10	EXTI LINE10	26	Reserved
11	EXTI LINE11	27	Reserved

Table 73. DMAMUX1: assignment of trigger inputs to resources (continued)

Trigger input	Resource	Trigger input	Resource
12	EXTI LINE12	28	Reserved
13	EXTI LINE13	29	Reserved
14	EXTI LINE14	30	Reserved
15	EXTI LINE15	31	Reserved

Table 74. DMAMUX1: assignment of synchronization inputs to resources

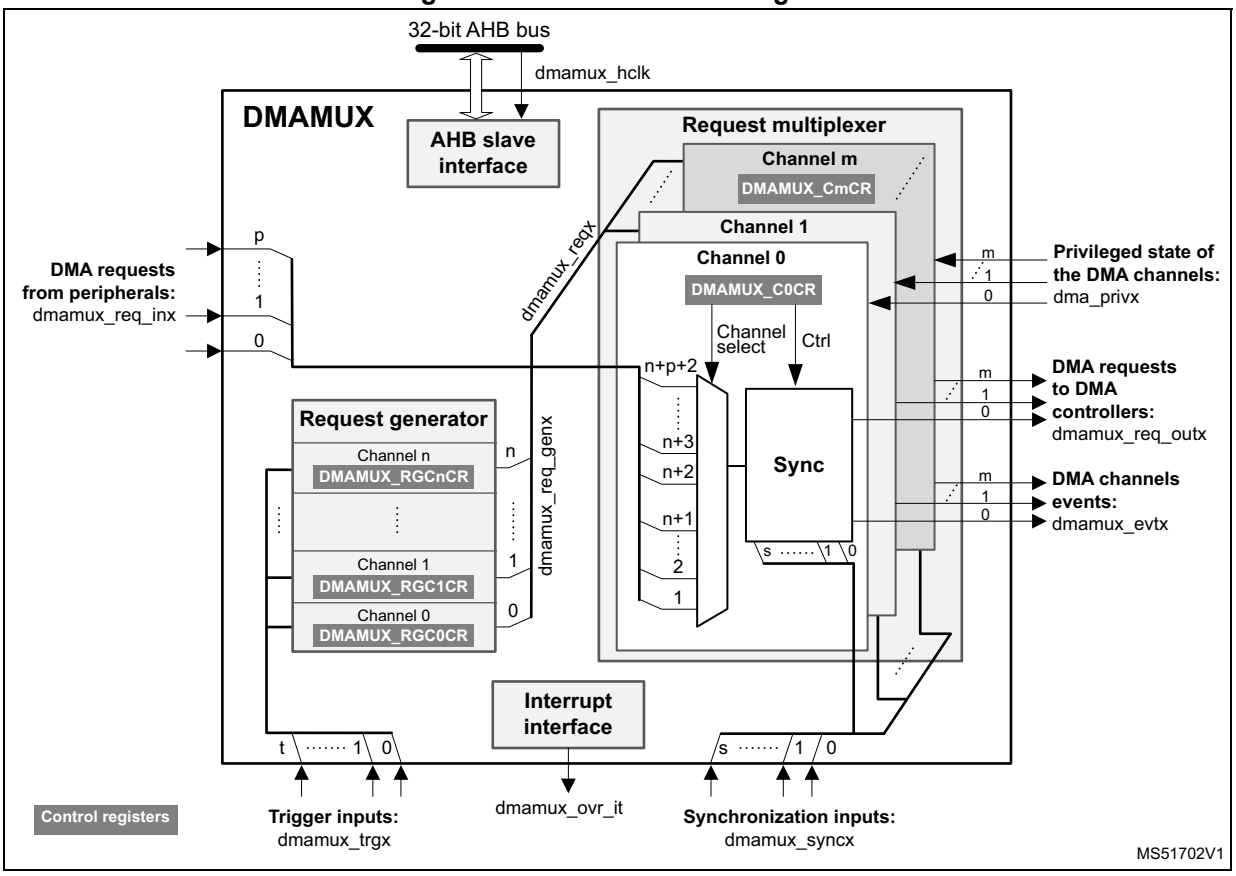
Sync. input	Resource	Sync. input	Resource
0	EXTI LINE0	16	dmamux_evt0
1	EXTI LINE1	17	dmamux_evt1
2	EXTI LINE2	18	LPTIM1_OUT
3	EXTI LINE3	19	LPTIM2_OUT
4	EXTI LINE4	20	LPTIM3_OUT
5	EXTI LINE5	21	Reserved
6	EXTI LINE6	22	Reserved
7	EXTI LINE7	23	Reserved
8	EXTI LINE8	24	Reserved
9	EXTI LINE9	25	Reserved
10	EXTI LINE10	26	Reserved
11	EXTI LINE11	27	Reserved
12	EXTI LINE12	28	Reserved
13	EXTI LINE13	29	Reserved
14	EXTI LINE14	30	Reserved
15	EXTI LINE15	31	Reserved

12.4 DMAMUX functional description

12.4.1 DMAMUX block diagram

Figure 37 shows the DMAMUX block diagram.

Figure 37. DMAMUX block diagram



DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (dmamux_reqx) from peripherals (dmamux_req_inx) and from channels of the DMAMUX request generator sub-block (dmamux_req_genx)
- DMAMUX request outputs to channels of DMA controllers (dmamux_req_outx)
- Internal or external signals to DMA request trigger inputs (dmamux_trgx)
- Internal or external signals to synchronization inputs (dmamux_syncx)

12.4.2 DMAMUX signals

Table 75 lists the DMAMUX signals.

Table 75. DMAMUX signals

Signal name	Description
dmamux_hclk	DMAMUX AHB clock
dmamux_req_inx	DMAMUX DMA request line inputs from peripherals
dmamux_trgx	DMAMUX DMA request triggers inputs (to request generator sub-block)
dmamux_req_genx	DMAMUX request generator sub-block channels outputs
dmamux_reqx	DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels)
dmamux_syncx	DMAMUX synchronization inputs (to request multiplexer sub-block)
dmamux_req_outx	DMAMUX requests outputs (to DMA controllers)
dma_privx	Privileged mode of each DMA controller request channel
dmamux_evtx	DMAMUX events outputs
dmamux_ovr_it	DMAMUX overrun interrupts

12.4.3 DMAMUX channels

A DMAMUX channel is a DMAMUX request multiplexer channel that may include, depending on the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to one single channel of DMA controller(s).

Channel configuration procedure

Follow the sequence below to configure both a DMAMUX x channel and the related DMA channel y:

1. Set to privileged or unprivileged the DMA channel y by a privileged write access to the privileged control bit of the DMA channel y configuration register.
2. Set and configure completely the DMA channel y, except enabling the channel y.
3. Set and configure completely the related DMAMUX y channel.
4. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

12.4.4 DMAMUX privileged / unprivileged channels

The DMAMUX is aware of the privileged or unprivileged state of a given DMA connected channel, and manages consequently its DMAMUX requested channel.

Note: A DMA controller(s) channel must be first configured as privileged or unprivileged, before the configuration of the connected DMAMUX channel.

Note: A privileged software is able to access any DMAMUX register, privileged or unprivileged. An unprivileged software is restricted to access only unprivileged DMAMUX register or register fields.

When a privileged software configures a DMA channel x either as privileged, an unprivileged software is not able to access (write is ignored, read returns zero) the related DMAMUX channel registers or register fields.

12.4.5 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced either from the peripherals or from the DMAMUX request generator.

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the DMAREQ_ID field in the DMAMUX_CxCR register.

Note: The null value in the field DMAREQ_ID corresponds to no DMA request line selected.

Caution: A same non-null DMAREQ_ID must not be programmed to different x and y DMAMUX request multiplexer channels (via DMAMUX_CxCR and DMAMUX_CyCR), except if application guarantees that the two connected DMA channels are not simultaneously active.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

Synchronization mode and channel event generation

Each DMAMUX request line multiplexer channel x can be individually synchronized by setting the synchronization enable (SE) bit in the DMAMUX_CxCR register.

DMAMUX has multiple synchronization inputs. The synchronization inputs are connected in parallel to all the channels of the request multiplexer.

The synchronization input is selected via the SYNC_ID field in the DMAMUX_CxCR register of a given channel x.

When a channel is in this synchronization mode, the selected input DMA request line is propagated to the multiplexer channel output, once is detected a programmable rising/falling edge on the selected input synchronization signal, via the SPOL[1:0] field of the DMAMUX_CxCR register.

Additionally, there is a programmable DMA request counter, internally to the DMAMUX request multiplexer, which may be used for the channel request output generation and also possibly for an event generation. An event generation on the channel x output is enabled through the EGE bit (event generation enable) of the DMAMUX_CxCR register.

As shown in *Figure 39*, upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

Note: If a synchronization event occurs while there is no pending selected input DMA request line, it is discarded. The following asserted input request lines is not connected to the DMAMUX multiplexer channel output until a synchronization event occurs again.

From this point on, each time the connected DMAMUX request is served by the DMA controller (a served request is deasserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in NBREQ field of the DMAMUX_CxCR register and the input DMA request line is disconnected from the multiplexer channel x output.

Thus, the number of DMA requests transferred to the multiplexer channel x output following a detected synchronization event, is equal to the value in NBREQ field, plus one.

Note: The NBREQ field value shall only be written by software when both synchronization enable bit SE and event generation enable EGE bit of the corresponding multiplexer channel x are disabled.

Figure 38. Synchronization mode of the DMAMUX request line multiplexer channel

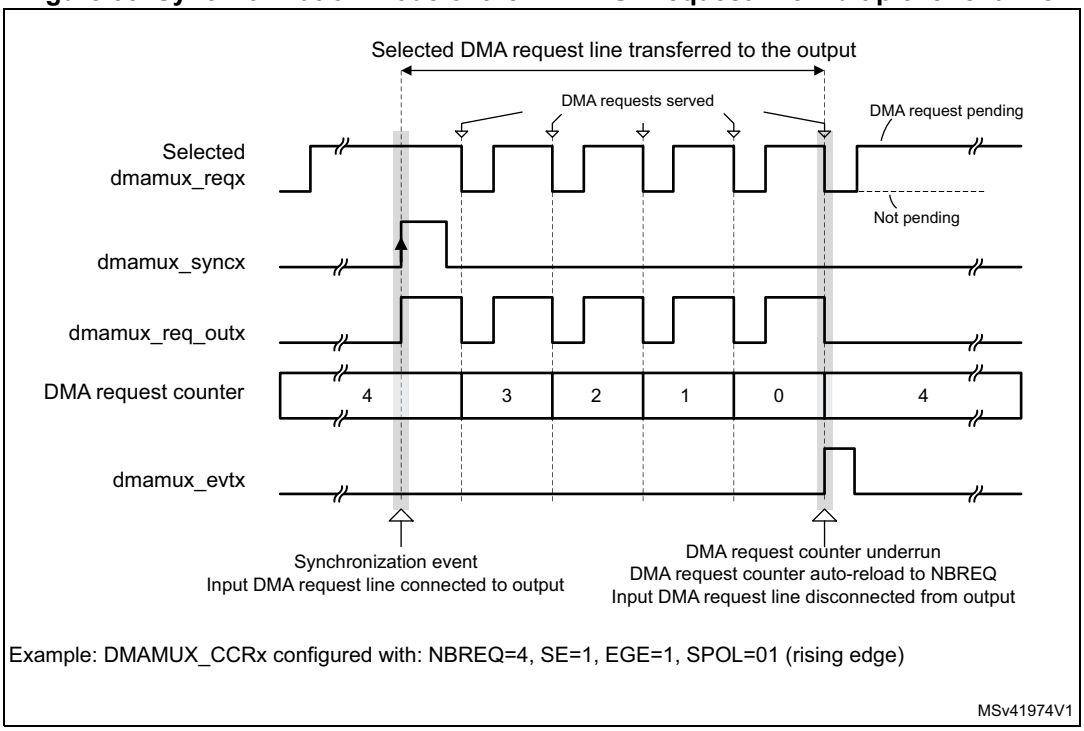
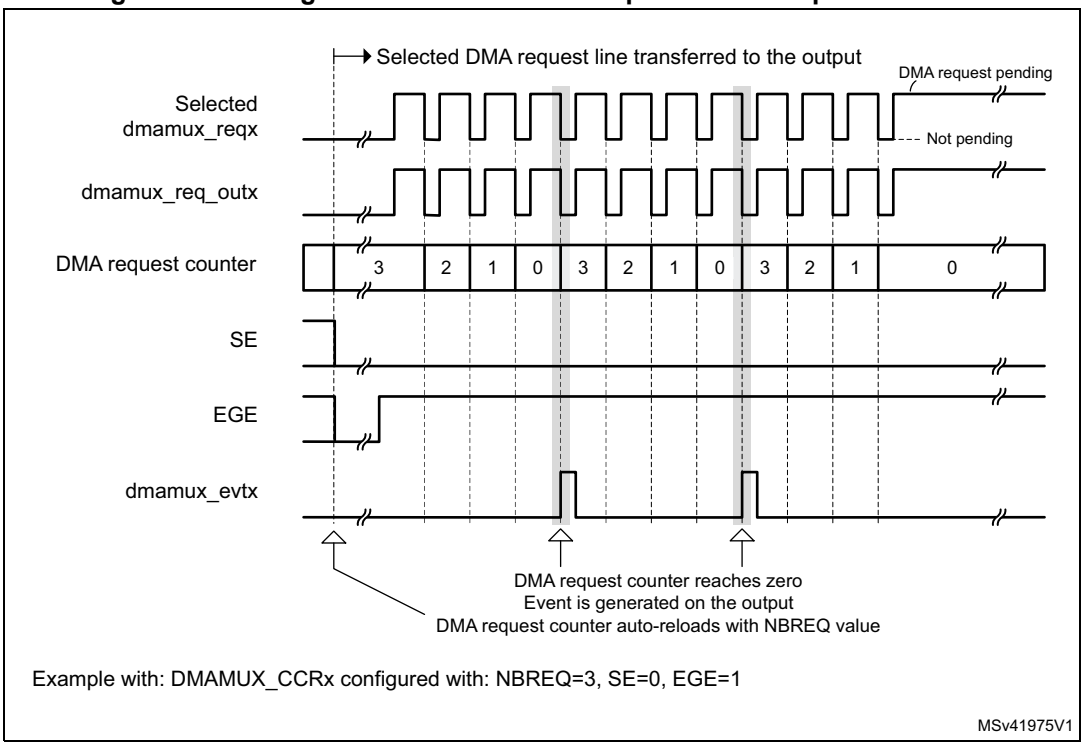


Figure 39. Event generation of the DMA request line multiplexer channel



If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in *Figure 38* and *Figure 39*.

Note: If EGE is enabled and NBREQ = 0, an event is generated after each served DMA request.

Note: A synchronization event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_CxCR register, the synchronization events are masked during three AHB clock cycles.

Synchronization overrun and interrupt

If a new synchronization event occurs before the request counter underrun (the internal request counter programmed via the NBREQ field of the DMAMUX_CxCR register), the synchronization overrun flag bit SOFx is set in the DMAMUX_CSR status register.

Note: The request multiplexer channel x synchronization must be disabled (DMAMUX_CxCR.SE = 0) at the completion of the use of the related channel of the DMA controller. Else, upon a new detected synchronization event, there is a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.

The overrun flag SOFx is reset by setting the associated clear synchronization overrun flag bit CSOFx in the DMAMUX_CFRDMAMUX_CCFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX_CxCR register.

12.4.6 DMAMUX request generator

The DMAMUX request generator produces DMA requests following trigger events on its DMA request trigger inputs.

The DMAMUX request generator has multiple channels. DMA request trigger inputs are connected in parallel to all channels.

The outputs of DMAMUX request generator channels are inputs to the DMAMUX request line multiplexer.

Each DMAMUX request generator channel x has an enable bit GE (generator enable) in the corresponding DMAMUX_RGxCR register.

The DMA request trigger input for the DMAMUX request generator channel x is selected through the SIG_ID (trigger signal ID) field in the corresponding DMAMUX_RGxCR register.

Trigger events on a DMA request trigger input can be rising edge, falling edge or either edge. The active edge is selected through the GPOL (generator polarity) field in the corresponding DMAMUX_RGxCR register.

Upon the trigger event, the corresponding generator channel starts generating DMA requests on its output. Each time the DMAMUX generated request is served by the connected DMA controller (a served request is deasserted), a built-in (inside the DMAMUX request generator) DMA request counter is decremented. At its underrun, the request generator channel stops generating DMA requests and the DMA request counter is automatically reloaded to its programmed value upon the next trigger event.

Thus, the number of DMA requests generated after the trigger event is GNBREQ + 1.

Note: The GNBREQ field value must be written by software only when the enable GE bit of the corresponding generator channel x is disabled.

There is no hardware write protection.

A trigger event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_RGxCR register, the trigger events are masked during three AHB clock cycles.

Trigger overrun and interrupt

If a new DMA request trigger event occurs before the DMAMUX request generator counter underrun (the internal counter programmed via the GNBREQ field of the DMAMUX_RGxCR register), and if the request generator channel x was enabled via GE, then the request trigger event overrun flag bit OFx is asserted by the hardware in the status DMAMUX_RGSR register.

Note: The request generator channel x must be disabled (DMAMUX_RGxCR.GE = 0) at the completion of the usage of the related channel of the DMA controller. Else, upon a new detected trigger event, there is a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.

The overrun flag OFx is reset by setting the associated clear overrun flag bit COFx in the DMAMUX_RGCFR register.

Setting the DMAMUX request trigger overrun flag generates an interrupt if the DMA request trigger event overrun interrupt enable bit OIE is set in the DMAMUX_RGxCR register.

12.5 DMAMUX interrupts

An interrupt can be generated upon:

- a synchronization event overrun in each DMA request line multiplexer channel
- a trigger event overrun in each DMA request generator channel

For each case, per-channel individual interrupt enable, status and clear flag register bits are available.

Table 76. DMAMUX interrupts

Interrupt signal	Interrupt event	Event flag	Clear bit	Enable bit
dmamuxovr_it	Synchronization event overrun on channel x of the DMAMUX request line multiplexer	SOFx	CSOFx	SOIE
	Trigger event overrun on channel x of the DMAMUX request generator	OFx	COFx	OIE

12.6 DMAMUX registers

Refer to the table containing register boundary addresses for the DMAMUX base address.

DMAMUX registers may be accessed per (8-bit) byte, (16-bit) half-word, or (32-bit) word. The address must be aligned with the data size.

12.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)

Address offset: 0x000 + 0x04 * x (x = 0 to 13)

Reset value: 0x0000 0000

This register shall be accessed by a privileged or unprivileged read/write, according to the privileged mode of the considered DMAMUX request line multiplexer channel x, depending on the privileged control bit of the connected of the connected DMA controller channel y. This assumes that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMUX mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL[1:0]		SE
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	DMAREQ_ID[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SYNC_ID[4:0]**: Synchronization identification

Selects the synchronization input (see).

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward

Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.

This field shall only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity

Defines the edge polarity of the selected synchronization input:

00: No event, i.e. no synchronization nor detection.

01: Rising edge

10: Falling edge

11: Rising and falling edges

Bit 16 **SE**: Synchronization enable

0: Synchronization disabled

1: Synchronization enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EGE**: Event generation enable

0: Event generation disabled

1: Event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bits 7:0 **DMAREQ_ID[7:0]**: DMA request identification
Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

12.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SOF13	SOF12	SOF11	SOF10	SOF9	SOF8	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **SOF[13:0]**: Synchronization overrun event flag
The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.
The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX_CFR DMAMUX_CCFR register.

12.6.3 DMAMUX request line multiplexer interrupt channel clear flag register (DMAMUX_CCFR)

Address offset: 0x084

Reset value: 0x0000 0000

This register must be written at bit level by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel x, depending on the privileged control bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMXUX mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CSOF 13	CSOF 12	CSOF 11	CSOF 10	CSOF 9	CSOF 8	CSOF 7	CSOF 6	CSOF 5	CSOF 4	CSOF 3	CSOF 2	CSOF 1	CSOF 0
		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **CSOF[13:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX_CSR register.

12.6.4 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)

Address offset: 0x100 + 0x04 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register shall be written by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (refer to DMAMUX_CyCR.DMAREQ_ID[7:0] and to the DMAMXUX mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL[1:0]		GE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	SIG_ID[4:0]				
							rw				rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ +1.

Note: This field must be written only when GE bit is disabled.

Bits 18:17 **GPOL[1:0]**: DMA request generator trigger polarity

Defines the edge polarity of the selected trigger input

00: No event, i.e. no trigger detection nor generation.

01: Rising edge

10: Falling edge

11: Rising and falling edges

Bit 16 **GE**: DMA request generator channel x enable

0: DMA request generator channel x disabled

1: DMA request generator channel x enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **OIE**: Trigger overrun interrupt enable

0: Interrupt on a trigger overrun event occurrence is disabled

1: Interrupt on a trigger overrun event occurrence is enabled

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **SIG_ID[4:0]**: Signal identification

Selects the DMA request trigger input used for the channel x of the DMA request generator

12.6.5 DMAMUX request generator interrupt status register (DMAMUX_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

This register shall be accessed at bit level by an unprivileged or privileged read, according to the privileged mode of the considered DMAMUX request line multiplexer channel x, depending on the privileged control bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMXUX mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF3	OF2	OF1	OF0
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **OF[3:0]**: Trigger overrun event flag

The flag is set when a new trigger event occurs on DMA request generator channel x, before the request counter underrun (the internal request counter programmed via the GNBREQ field of the DMAMUX_RGxCR register).

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX_RGCFR register.

12.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)

Address offset: 0x144

Reset value: 0x0000 0000

This register shall be written at bit level by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (refer to DMAMUX_CyCR.DMAREQ_ID[7:0] and to the DMAMXUX mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF3	COF2	COF1	COF0
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **COF[3:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX_RGSR register.

12.6.7 DMAMUX register map

The following table summarizes the DMAMUX registers and reset values. Refer to the register boundary address table for the DMAMUX register base address.

Table 77. DMAMUX register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMAMUX_C0CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x004	DMAMUX_C1CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x008	DMAMUX_C2CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x00C	DMAMUX_C3CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x010	DMAMUX_C4CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x014	DMAMUX_C5CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x018	DMAMUX_C6CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x01C	DMAMUX_C7CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x020	DMAMUX_C8CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x024	DMAMUX_C9CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x028	DMAMUX_C10CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x02C	DMAMUX_C11CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x030	DMAMUX_C12CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x034	DMAMUX_C13CR	Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x038 - 0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x080	DMAMUX_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOF13	SOF12	SOF11	SOF10	SOF9	SOF8	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DMAMUX_CCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSOF13	CSOF12	CSOF11	CSOF10	CSOF9	CSOF8	CSOF7	CSOF6	CSOF5	CSOF4	CSOF3	CSOF2	CSOF1	CSOF0
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 77. DMAMUX register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088 - 0x0FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x100	DMAMUX_RG0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL[1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	0	0								0				0	0	0	0	0
0x104	DMAMUX_RG1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL[1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	0	0								0				0	0	0	0	0
0x108	DMAMUX_RG2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL[1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	0	0								0				0	0	0	0	0
0x10C	DMAMUX_RG3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL[1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	0	0								0				0	0	0	0	0
0x110 - 0x13C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x140	DMAMUX_RGSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF3	OF2	OF1	OF0
	Reset value																													0	0	0	0
0x144	DMAMUX_RGCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF3	COF2	COF1	COF0
	Reset value																													0	0	0	0
0x148 - 0x3FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.4 on page 62* for the register boundary addresses.

13 Nested vectored interrupt controller (NVIC)

13.1 NVIC main features

- 62 maskable interrupt channels (not including the sixteen Cortex-M4 with DSP interrupt lines)
- 16 programmable priority levels (four bits of interrupt priority used)
- Low-latency exception interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interfaces are closely coupled, resulting in low-latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC.

For more information on exceptions and NVIC programming, refer to the PM0214 programming manual for Cortex®-M4 (PM0214).

13.2 Interrupt and exception vectors

The vector table is given in *Table 78* (shaded cells indicate the processor exceptions).

Table 78. Vector table

Position	Priority	Type of priority	Acronym	Description(1)	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	Fixed	Reset	Reset	0x0000 0004
-	-2	Fixed	NMI	Non maskable interrupt HSE32 CSS, Flash ECC and SRAM2 parity	0x0000 0008
-	-1	Fixed	HardFault	All classes of fault	0x0000 000C
-	0	Settable	MemManager	Memory manager	0x0000 0010
-	1	Settable	BusFault	Prefetch fault, memory access fault	0x0000 0014
-	2	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C 0x0000 0028
-	3	Settable	SVCall	System service can via SWI instruction	0x0000 002C
-	4	Settable	Debug	Debug monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	5	Settable	PendSV	Pendable request for system service	0x0000 0038
-	6	Settable	SysTick	SysTick timer	0x0000 003C
0	7	Settable	WWDG	Window watchdog early wakeup	0x0000 0040

Table 78. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description[1]	Address
1	8	Settable	PVD, PVM[3]	PVD through EXTI[16] PVM[3] through EXTI[34]	0x0000 0044
2	9	Settable	TAMP, RTC_STAMP, LSE_CSS, RTC_SSRU	TAMP tamper RTC timestamp LSECSS interrupt RTC SSR underflow interrupt	0x0000 0048
3	10	Settable	RTC_WKUP	RTC wakeup interrupt	0x0000 004C
4	11	Settable	FLASH	Flash memory global interrupt and Flash memory ECC single error interrupt	0x0000 0050
5	12	Settable	RCC	RCC global interrupt	0x0000 0054
6	13	Settable	EXTI0	EXTI line 0 interrupt through EXTI[0]	0x0000 0058
7	14	Settable	EXTI1	EXTI line 1 interrupt through EXTI[1]	0x0000 005C
8	15	Settable	EXTI2	EXTI line 2 interrupt through EXTI[2]	0x0000 0060
9	16	Settable	EXTI3	EXTI line 3 interrupt through EXTI[3]	0x0000 0064
10	17	Settable	EXTI4	EXTI line 4 interrupt through EXTI[4]	0x0000 0068
11	18	Settable	DMA1_CH1	DMA1 channel 1 non-secure interrupt	0x0000 006C
12	19	Settable	DMA1_CH2	DMA1 channel 2 non-secure interrupt	0x0000 0070
13	20	Settable	DMA1_CH3	DMA1 channel 3 non-secure interrupt	0x0000 0074
14	21	Settable	DMA1_CH4	DMA1 channel 4 non-secure interrupt	0x0000 0078
15	22	Settable	DMA1_CH5	DMA1 channel 5 non-secure interrupt	0x0000 007C
16	23	Settable	DMA1_CH6	DMA1 channel 6 non-secure interrupt	0x0000 0080
17	24	Settable	DMA1_CH7	DMA1 channel 7 non-secure interrupt	0x0000 0084
18	25	Settable	ADC	ADC global interrupt	0x0000 0088
19	26	Settable	DAC	DAC global interrupt	0x0000 008C
20	27	Settable	Reserved	Reserved	0x0000 0090
21	28	Settable	COMP	COMP2 and COMP1 interrupt through EXTI[22:21]	0x0000 0094
22	29	Settable	EXTI[9:5]	EXTI line [9:5] interrupt through EXTI[9:5]	0x0000 0098
23	30	Settable	TIM1_BRK	Timer 1 break interrupt	0x0000 009C
24	31	Settable	TIM1_UP	Timer 1 Update	0x0000 00A0
25	32	Settable	TIM1_TRG_COM	Timer 1 trigger and communication	0x0000 00A4
26	33	Settable	TIM1_CC	Timer 1 capture compare interrupt	0x0000 00A8
27	34	Settable	TIM2	Timer 2 global interrupt	0x0000 00AC
28	35	Settable	TIM16	Timer 16 global interrupt	0x0000 00B0
29	36	Settable	TIM17	Timer 17 global interrupt	0x0000 00B4
30	37	Settable	I2C1_EV	I2C1 event interrupt	0x0000 00B8

Table 78. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description[1]	Address
31	38	Settable	I2C1_ER	I2C1 error interrupt	0x0000 00BC
32	39	Settable	I2C2_EV	I2C2 event interrupt	0x0000 00C0
33	40	Settable	I2C2_ER	I2C2 error interrupt	0x0000 00C4
34	41	Settable	SPI1	SPI1 global interrupt	0x0000 00C8
35	42	Settable	SPI2S2	SPI2S2 global interrupt	0x0000 00CC
36	43	Settable	USART1	USART1 global interrupt	0x0000 00D0
37	44	Settable	USART2	USART2 global interrupt	0x0000 00D4
38	45	Settable	LPUART1	LPUART1 global interrupt	0x0000 00D8
39	46	Settable	LPTIM1	LP timer 1 global interrupt	0x0000 00DC
40	47	Settable	LPTIM2	LP timer 2 global interrupt	0x0000 00E0
41	48	Settable	EXTI[15:10]	EXTI line [15:10] interrupt through EXTI[15:10]	0x0000 00E4
42	49	Settable	RTC_ALARM	RTC alarms A and B interrupt	0x0000 00E8
43	50	Settable	LPTIM3	LP timer 3 global interrupt	0x0000 00EC
44	51	Settable	Reserved	Reserved	0x0000 00F0
45	52	Settable	Reserved	Reserved	0x0000 00F4
46	53	Settable	Reserved	Reserved	0x0000 00F8
47	54	Settable	HSEM	Semaphore interrupt 0 to CPU	0x0000 00FC
48	55	Settable	I2C3_EV	I2C3 event interrupt	0x0000 0100
49	56	Settable	I2C3_ER	I2C3 error interrupt	0x0000 0104
50	57	Settable	Radio IRQ, Busy	Radio IRQs RFBUSY interrupt through EXTI[45]	0x0000 0108
51	58	Settable	AES	AES global interrupt	0x0000 010C
52	59	Settable	True RNG	True random number generator interrupt	0x0000 0110
53	60	Settable	PKA	Private key accelerator interrupt	0x0000 0114
54	61	Settable	DMA2_CH1	DMA2 channel 1 non-secure interrupt	0x0000 0118
55	62	Settable	DMA2_CH2	DMA2 channel 2 non-secure interrupt	0x0000 011C
56	63	Settable	DMA2_CH3	DMA2 channel 3 non-secure interrupt	0x0000 0120
57	64	Settable	DMA2_CH4	DMA2 channel 4 non-secure interrupt	0x0000 0124
58	65	Settable	DMA2_CH5	DMA2 channel 5 non-secure interrupt	0x0000 0128
59	66	Settable	DMA2_CH6	DMA2 channel 6 non-secure interrupt	0x0000 012C
60	67	Settable	DMA2_CH7	DMA2 channel 7 non-secure interrupt	0x0000 0130
61	68	Settable	DMAMUX1_OVR	DMAMUX1 overrun interrupt	0x0000 0134

1. EXTI[n] refer to the input event number [n] of the EXTI.

14 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wakeup through configurable and direct event inputs. It provides wakeup requests to the power control and generates an interrupt request to the CPU NVIC and events to the CPU event input.

For the CPU, an additional event Generation block (EVG) is needed to generate the CPU event signal.

The EXTI wakeup requests allow the system to be woken up from Stop modes and the CPU to be woken up from the CStop and CStandby modes.

The interrupt request and event request generation can also be used in Run modes.

14.1 EXTI main features

The EXTI main features are the following:

- 47 input events are supported.
- All event inputs allow the system to be waked up.
- Events which do not have an associated wakeup flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI.

The asynchronous event inputs are classified in the following two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse), with features listed below:
  - Selectable active trigger edge
  - Interrupt pending status register bit
  - Individual interrupt and event generation mask, used for conditioning the CPU wakeup, interrupt and event generation
  - SW trigger possibility
- Direct events (interrupt and wakeup sources from peripherals, having an associated flag to be cleared in the peripheral), with features listed below:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (The interrupt pending status flag is provided by the peripheral generating the event.)
  - Individual interrupt and event generation mask, used for conditioning the CPU wakeup and event generation
  - No software trigger possibility

14.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block and the masking block as shown in *Figure 40*.

The register block contains all the EXTI registers.

The event input trigger block provides event input edge trigger logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, and the masking of these.

Figure 40. EXTI block diagram

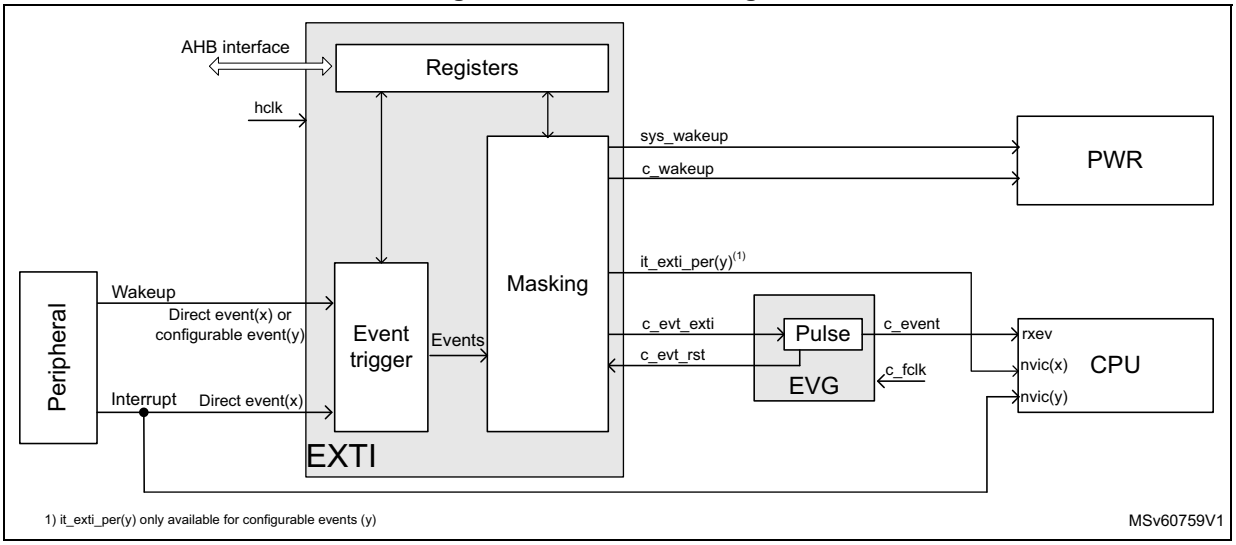


Table 79. EXTI pin overview

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wakeup events from peripherals which do not have an associated interrupt and flag in the peripheral
Direct event(x)	I	Synchronous and asynchronous wakeup events from peripherals which have an associated interrupt and flag in the peripheral
it_exti_per (y)	O	Interrupts to the CPU associated with the configurable event (y)
c_evt_exti	O	High-level sensitive event output for the CPU, synchronous to hclk
c_evt_rst	I	Asynchronous reset input to clear c_evt_exti
sys_wakeup	O	Asynchronous system wakeup request to PWR for ck_sys and hclk
c_wakeup	O	Wakeup request to PWR for the CPU, synchronous to hclk

Table 80. EVG pin overview

Pin name	I/O	Description
c_fclk	I	CPU free running clock
c_evt_in	I	High-level sensitive events input from EXTI, asynchronous to the CPU clock
c_event	O	Event pulse, synchronous to the CPU clock
c_evt_rst	O	Event reset signal, synchronous to the CPU clock

14.3 EXTI connections between peripherals and CPU

The peripherals able to generate wakeup or interrupt events when the system is in Stop mode, are connected to the EXTI.

Peripheral wakeup signals that generate a pulse or that do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit which requires to be cleared. It is the EXTI interrupt associated with the status bit that interrupts the CPU.

Peripheral interrupt and wakeup signals that have a status bit in the peripheral which requires to be cleared in the peripheral, are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wakeup is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.

The EXTI configurable event interrupts are connected to the NVIC of the CPU.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wakeup signalis connected to the PWR block and are used to wake up the system and CPU sub-system bus clocks.

14.3.1 EXTI wakeup interrupt list

The wakeup sources are listed in *Table 81: Wakeup interrupts*.

Some wakeup sources are able to generate an event to the CPU (see ‘Event’ column).

For CPU interrupt handling, see *Section 13: Nested vectored interrupt controller (NVIC)*.

Table 81. Wakeup interrupts

EXTI n°	Acronym	Description	EXTI type	Event	Wakeup
0	EXTI[0]	EXTI line 0 from SYSCFG	Configurable	Yes	CPU
1	EXTI[1]	EXTI line 1 from SYSCFG	Configurable	Yes	CPU
2	EXTI[2]	EXTI line 2 from SYSCFG	Configurable	Yes	CPU
3	EXTI[3]	EXTI line 3 from SYSCFG	Configurable	Yes	CPU
4	EXTI[4]	EXTI line 4 from SYSCFG	Configurable	Yes	CPU
5	EXTI[5]	EXTI line 5 from SYSCFG	Configurable	Yes	CPU
6	EXTI[6]	EXTI line 6 from SYSCFG	Configurable	Yes	CPU
7	EXTI[7]	EXTI line 7 from SYSCFG	Configurable	Yes	CPU
8	EXTI[8]	EXTI line 8 from SYSCFG	Configurable	Yes	CPU
9	EXTI[8]	EXTI line 9 from SYSCFG	Configurable	Yes	CPU
10	EXTI[10]	EXTI line 10 from SYSCFG	Configurable	Yes	CPU
11	EXTI[11]	EXTI line 11 from SYSCFG	Configurable	Yes	CPU
12	EXTI[12]	EXTI line 12 from SYSCFG	Configurable	Yes	CPU
13	EXTI[13]	EXTI line 13 from SYSCFG	Configurable	Yes	CPU
14	EXTI[14]	EXTI line 14 from SYSCFG	Configurable	Yes	CPU

Table 81. Wakeup interrupts (continued)

EXTI n°	Acronym	Description	EXTI type	Event	Wakeup
15	EXTI[15]	EXTI line 15 from SYSCFG	Configurable	Yes	CPU
16	PVD	PVD line	Configurable	No	CPU
17	RTC_ALARM	RTC alarms A and B interrupt	Direct	Yes	CPU
18	SSRU	RTC SSR underflow interrupt	Direct	Yes	CPU
19	TAMP, RTC_STAMP, LSE_CSS	TAMP tamper interrupt RTC timestamp interrupt RCC LSECSS interrupt	Direct	Yes	CPU
20	RTC_WKUP	RTC wakeup interrupt	Direct	Yes	CPU
21	COMP1	COMP1 line	Configurable	Yes	CPU
22	COMP2	COMP2 line	Configurable	Yes	CPU
23	I2C1 wakeup	I2C1 wakeup	Direct	No	CPU
24	I2C2 wakeup	I2C2 wakeup	Direct	No	CPU
25	I2C3 wakeup	I2C3 wakeup	Direct	No	CPU
26	USART1	USART1 wakeup	Direct	No	CPU
27	USART2	USART2 wakeup	Direct	No	CPU
28	LPUART1	LPUART1 wakeup	Direct	No	CPU
29	LPTIM1 wakeup	LPtimer 1 wakeup	Direct	No	CPU
30	LPTIM2 wakeup	LPtimer 2 wakeup	Direct	No	CPU
31	LPTIM3 wakeup	LPtimer 3 wakeup	Direct	No	CPU
32	Reserved	-	Direct	No	-
33	Reserved	-	Direct	No	-
34	PVM[3]	PVM[3] line	Configurable	No	CPU
35	Reserved	-	Direct	No	-
36	Reserved	-	Direct	No	-
37	Reserved	-	Direct	No	-
38	HSEM interrupt 0	Semaphore interrupt 0 with CPU	Direct	No	CPU
39	Reserved	-	Direct	No	-
40	Reserved	-	Configurable	No	-
41	Reserved	-	Configurable	No	-
42	Flash	Flash ECC and global interrupts	Direct	No	CPU
43	HSE32 CSS interrupt	RCC HSE32 CSS interrupt	Direct	No	CPU
44	Radio IRQs	Radio IRQs interrupts	Direct	No	CPU
45	Radio Busy	RFBUSY wakeup	Configurable	No	CPU
46	CDBGPWRUPREQ	Debug power-up request wakeup	Direct	No	CPU

14.4 EXTI functional description

Depending on the EXTI event input type and wakeup targets, different logic implementations are used. The applicable features are controlled from register bits as detailed below:

- Active trigger edge enable
  - by rising edge selection
    EXTI rising trigger selection register (EXTI_RTSR1)
    EXTI rising trigger selection register (EXTI_RTSR2)
  - by falling edge selection
    EXTI falling trigger selection register (EXTI_FTSR1)
    EXTI falling trigger selection register (EXTI_FTSR2)
- Software trigger
  EXTI software interrupt event register (EXTI_SWIER1)
  EXTI software interrupt event register (EXTI_SWIER2)
- Interrupt pending flag
  EXTI pending register (EXTI_PR1)
  EXTI pending register (EXTI_PR2)
- CPU wakeup and interrupt enable
  EXTI interrupt mask register (EXTI_IMR1)
  EXTI interrupt mask register (EXTI_IMR2)
- CPU wakeup and event enable
  EXTI event mask register (EXTI_EMR1)

Table 82. EXTI event input configurations and register control

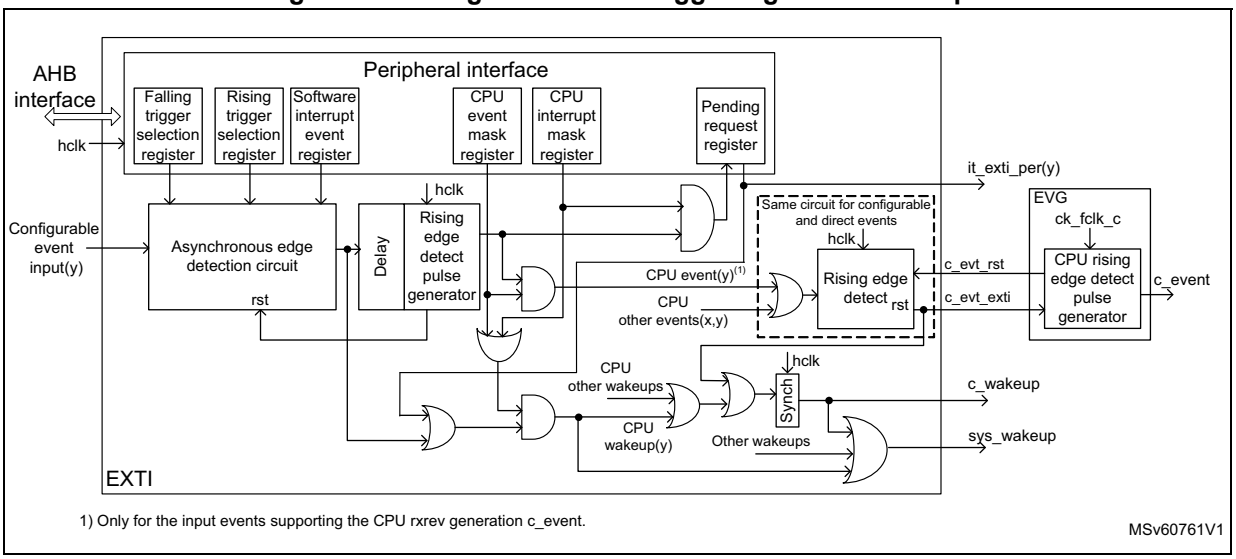
Event input type	Logic implementation	EXTI_ RTSR	EXTI_ FTSR	EXTI_ SWIER	EXTI_ PR	EXTI_ IMR	EXTI_ EMR(1)
Configurable	Configurable event input wakeup logic	x	x	x	x	x	x
Direct	Direct event input wakeup logic	-	-	-	-	x	x

1. Only for input events with configuration “rxev generation” enabled.

14.4.1 EXTI configurable event input wakeup

The extended interrupt/event block diagram for configurable events is shown in *Figure 41*. The configurable events allow the system and CPU wakeup from Sleep and Stop modes, and provide a pending flag in the EXTI.

Figure 41. Configurable event trigger logic CPU wakeup



The software interrupt event register allows configurable events to be triggered by software, writing the corresponding register bit, irrespective of the edge selection setting.

The rising and falling edges selection registers allow the configurable event active trigger edge (or both edges) to be enabled.

The CPU has its dedicated interrupt mask and event mask registers. The enabled event allows the generation of an event on the CPU. All events for a CPU are ORed together into a single CPU event signal. The event pending register (EXTI_PR) is not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers, shared by the CPU. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts need to be acknowledged by software in the EXTI_PR register.

When a CPU interrupt or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

Note: A detected configurable event interrupt pending request may be cleared by the CPU. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

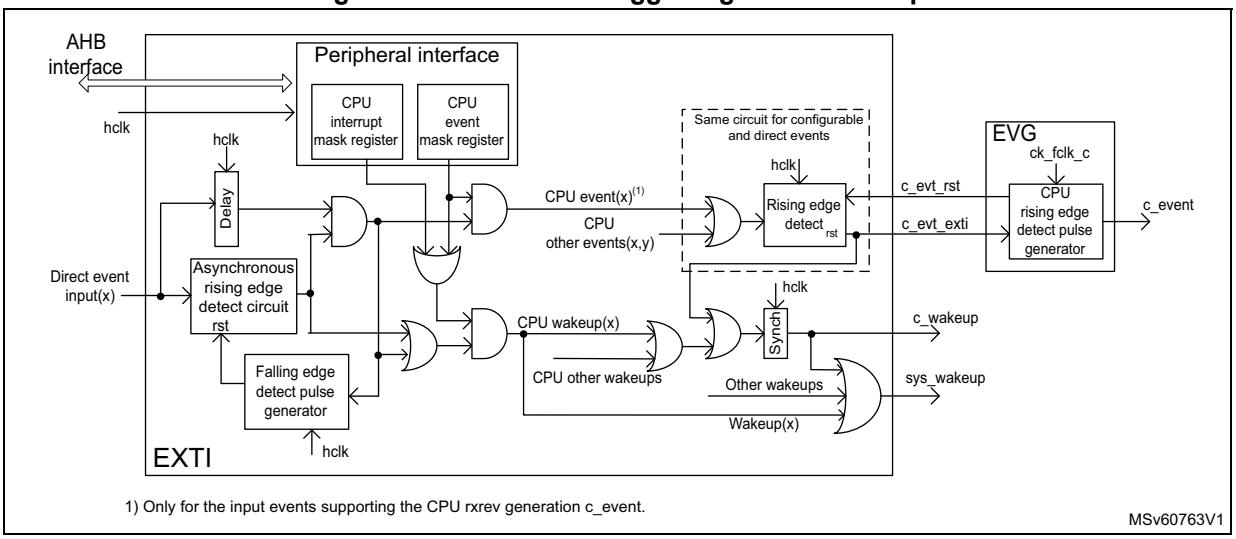
14.4.2 EXTI direct event input wakeup

The extended interrupt/event block diagram for direct events is shown in *Figure 42. T*he direct events allow the wakeup of the system and of the CPU from Sleep and Stop modes.

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU sub-system clocks and may generate a CPU wakeup event. The peripheral synchronous interrupt associated with the direct wakeup event, wake up the CPU. The EXTI direct event is able to generate a CPU event. This CPU event wakes up the CPU.

The CPU event may occur before the associated peripheral interrupt flag is set.

Figure 42. Direct event trigger logic CPU wakeup



14.5 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wakeup event. The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the generation of a CPU wakeup is conditioned by the CPU interrupt mask and the CPU event mask.

Table 83. Masking functionality

CPU interrupt enable EXTI_IMRm.IMb	CPU event enable EXTI_EMRm.EMb	Configurable event inputs EXTI_PRm.PIFb	it_exti_per(y)	CPU event	CPU wakeup
0	0	No	Masked	Masked	Masked
	1	No	Masked	Yes	Yes
1	0	Status latched	Yes	Masked	Yes(1)
	1	Status latched	Yes	Yes	Yes

1. Only if the CPU interrupt is enabled in EXTI_IMRm.IMb.

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated it_exti_per(y) interrupt is unmasked, the corresponding pending bit in EXTI_PR is set, the CPU sub-system wakes up and the CPU interrupt signal is activated. The EXTI_PR pending bit must be set to 1 by software. This clears the it_exti_per(y) interrupt.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit in the EXTI. When the associated direct event is unmasked in EXTI_IM, the CPU sub-system wakes up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU event must be unmasked in EXTI_EMR to generate an event. When the enabled edges occur on the event input, a CPU event pulse is generated. There is no event pending bit.

For the configurable event inputs, an event request can be generated by software, setting to 1 the corresponding bit in the interrupt/event register EXTI_SWIER. This allows the generation of a rising edge on the event. The edge event pending bit must be set in EXTI_PR, irrespective of the setting in EXTI_RTSR.

14.6 EXTI registers

The EXTI register map is divided in sections listed in the table below.

Table 84. EXTI register map sections

Address	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [63:32] configuration
0x080 - 0x0BC	CPU input event configuration

All these registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

14.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	Res.	Res.	Res.	Res.	RT16
									rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **RT22:** rising trigger event configuration bit of configurable event input 22

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

Note: The configurable event inputs are edge triggered. No glitch must be generated on these inputs. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bit 21 **RT21:** rising trigger event configuration bit of configurable event input 21

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **RT16:** rising trigger event configuration bit of configurable event input 16

Bit 15 **RT15:** rising trigger event configuration bit of configurable event input 15

Bit 14 **RT14:** rising trigger event configuration bit of configurable event input 14

Bit 13 **RT13:** rising trigger event configuration bit of configurable event input 13

Bit 12 **RT12:** rising trigger event configuration bit of configurable event input 12

Bit 11 **RT11:** rising trigger event configuration bit of configurable event input 11

Bit 10 **RT10:** rising trigger event configuration bit of configurable event input 10

Bit 9 **RT9:** rising trigger event configuration bit of configurable event input 9

Bit 8 **RT8:** rising trigger event configuration bit of configurable event input 8

Bit 7 **RT7:** rising trigger event configuration bit of configurable event input 7

Bit 6 **RT6:** rising trigger event configuration bit of configurable event input 6

Bit 5 **RT5:** rising trigger event configuration bit of configurable event input 5

Bit 4 **RT4:** rising trigger event configuration bit of configurable event input 4

Bit 3 **RT3:** rising trigger event configuration bit of configurable event input 3

Bit 2 **RT2:** rising trigger event configuration bit of configurable event input 2

Bit 1 **RT1:** rising trigger event configuration bit of configurable event input 1

Bit 0 **RT0:** rising trigger event configuration bit of configurable event input 0

14.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	Res.	Res.	Res.	Res.	FT16
									rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **FT22:** falling trigger event configuration bit of configurable event input 22

0: falling trigger disabled (for event and interrupt) for input line

1: falling trigger enabled (for event and interrupt) for input line

Note: The configurable event inputs are edge triggered. No glitch must be generated on these inputs. If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bit 21 **FT21:** falling trigger event configuration bit of configurable event input 21

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **FT16:** falling trigger event configuration bit of configurable event input 16

Bit 15 **FT15:** falling trigger event configuration bit of configurable event input 15

Bit 14 **FT14:** falling trigger event configuration bit of configurable event input 14

Bit 13 **FT13:** falling trigger event configuration bit of configurable event input 13

Bit 12 **FT12:** falling trigger event configuration bit of configurable event input 12

Bit 11 **FT11:** falling trigger event configuration bit of configurable event input 11

Bit 10 **FT10:** falling trigger event configuration bit of configurable event input 10

Bit 9 **FT9:** falling trigger event configuration bit of configurable event input 9

Bit 8 **FT8:** falling trigger event configuration bit of configurable event input 8

Bit 7 **FT7:** falling trigger event configuration bit of configurable event input 7

Bit 6 **FT6:** falling trigger event configuration bit of configurable event input 6

Bit 5 **FT5:** falling trigger event configuration bit of configurable event input 5

Bit 4 **FT4:** falling trigger event configuration bit of configurable event input 4

Bit 3 **FT3:** falling trigger event configuration bit of configurable event input 3

Bit 2 **FT2:** falling trigger event configuration bit of configurable event input 2

Bit 1 **FT1:** falling trigger event configuration bit of configurable event input 1

Bit 0 **FT0:** falling trigger event configuration bit of configurable event input 0

14.6.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI22	SWI21	Res.	Res.	Res.	Res.	SWI16
									rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SWI22:** Software interrupt on line 22

A software interrupt is generated independently from the setting in EXTI_RTSR and EXTI_FTSR. This bit always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 to this bit triggers an event on line 22.
This bit is automatically cleared by hardware.

Bit 21 **SWI21:** Software interrupt on line 21

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **SWI16:** Software interrupt on line 16

Bit 15 **SWI15:** Software interrupt on line 15

Bit 14 **SWI14:** Software interrupt on line 14

Bit 13 **SWI13:** Software interrupt on line 13

Bit 12 **SWI12:** Software interrupt on line 12

Bit 11 **SWI11:** Software interrupt on line 11

Bit 10 **SWI10:** Software interrupt on line 10

Bit 9 **SWI9:** Software interrupt on line 9

Bit 8 **SWI8:** Software interrupt on line 8

Bit 7 **SWI7:** Software interrupt on line 7

Bit 6 **SWI6:** Software interrupt on line 6

Bit 5 **SWI5:** Software interrupt on line 5

Bit 4 **SWI4:** Software interrupt on line 4

Bit 3 **SWI3:** Software interrupt on line 3

Bit 2 **SWI2:** Software interrupt on line 2

Bit 1 **SWI1:** Software interrupt on line 1

Bit 0 **SWI0:** Software interrupt on line 0

14.6.4 EXTI pending register (EXTI_PR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF22	PIF21	Res.	Res.	Res.	Res.	PIF16
									rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIF15	PIF14	PIF13	PIF12	PIF11	PIF10	PIF9	PIF8	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PIF22:** pending bit on event input 22

These bits are set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

0: No trigger request occurred.

1: Trigger request occurred.

Bit 21 **PIF21:** pending bit on event input 21

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **PIF16:** pending bit on event input 16

Bit 15 **PIF15:** pending bit on event input 15

Bit 14 **PIF14:** pending bit on event input 14

Bit 13 **PIF13:** pending bit on event input 13

Bit 12 **PIF12:** pending bit on event input 12

Bit 11 **PIF11:** pending bit on event input 11

Bit 10 **PIF10:** pending bit on event input 10

Bit 9 **PIF9:** pending bit on event input 9

Bit 8 **PIF8:** pending bit on event input 8

Bit 7 **PIF7:** pending bit on event input 7

Bit 6 **PIF6:** pending bit on event input 6

Bit 5 **PIF5:** pending bit on event input 5

Bit 4 **PIF4:** pending bit on event input 4

Bit 3 **PIF3:** pending bit on event input 3

Bit 2 **PIF2:** pending bit on event input 2

Bit 1 **PIF1:** pending bit on event input 1

Bit 0 **PIF0:** pending bit on event input 0

14.6.5 EXTI rising trigger selection register (EXTI_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RT45	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT34	Res.	Res.
		rw											rw		

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **RT45:** rising trigger event configuration bit of configurable event input 45

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

Note: The configurable event inputs are edge triggered. No glitch must be generated on these inputs. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 12:3 Reserved, must be kept at reset value.

Bit 2 **RT34:** rising trigger event configuration bit of configurable event input 34

Bits 1:0 Reserved, must be kept at reset value.

14.6.6 EXTI falling trigger selection register (EXTI_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FT45	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT34	Res.	Res.
		rw											rw		

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FT45:** falling trigger event configuration bit of configurable event input 45

0: Falling trigger disabled (for event and interrupt) for input line

1: Falling trigger enabled (for event and interrupt) for input line

Note: The configurable event inputs are edge triggered. No glitch must be generated on these inputs. If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 12:3 Reserved, must be kept at reset value.

Bit 2 **FT34:** falling trigger event configuration bit of configurable event input 34

Bits 1:0 Reserved, must be kept at reset value.

14.6.7 EXTI software interrupt event register (EXTI_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SWI45	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI34	Res.	Res.
		rw											rw		

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SWI45:** software interrupt on event 45
A software interrupt is generated independently from the setting in EXTI_RTSR and EXTI_FTSR. This bit always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 to this bit triggers an event on line 45.
This bit is automatically cleared by hardware.

Bits 12:3 Reserved, must be kept at reset value.

Bit 2 **SWI34:** software interrupt on event 34

Bits 1:0 Reserved, must be kept at reset value.

14.6.8 EXTI pending register (EXTI_PR2)

Address offset: 0x02C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PIF45	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF34	Res.	Res.
		rw											rw		

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **PIF45:** pending bit on event input 45
These bits are set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.
0: No trigger request occurred.
1: Trigger request occurred.

Bits 12:3 Reserved, must be kept at reset value.

Bit 2 **PIF34:** pending bit on event input 34

Bits 1:0 Reserved, must be kept at reset value.

14.6.9 EXTI interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IM[31:0]:** wakeup with interrupt mask on event input x (x= 31 to 0)
For each bit of this field:
0: Wakeup with interrupt request from line x is masked.
1: Wakeup with Interrupt request from line x is unmasked.

14.6.10 EXTI event mask register (EXTI_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM22	EM21	EM20	EM19	EM18	EM17	Res.
									rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **EM22:** wakeup with event generation mask on event input 22
0: Event request from line 22 is masked.
1: Event request from line 22 is unmasked.

Bit 21 **EM21:** wakeup with event generation mask on event input 21

Bit 20 **EM20:** wakeup with event generation mask on event input 20

Bit 19 **EM19:** wakeup with event generation mask on event input 19

Bit 18 **EM18:** wakeup with event generation mask on event input 18

Bit 17 **EM17:** wakeup with event generation mask on event input 17

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EM15:** wakeup with event generation mask on event input 15

Bit 14 **EM14:** wakeup with event generation mask on event input 14

Bit 13 **EM13:** wakeup with event generation mask on event input 13

Bit 12 **EM12:** wakeup with event generation mask on event input 12

Bit 11 **EM11:** wakeup with event generation mask on event input 11

Bit 10 **EM10:** wakeup with event generation mask on event input 10

Bit 9 **EM9:** wakeup with event generation mask on event input 19

Bit 8 **EM8:** wakeup with event generation mask on event input 8

Bit 7 **EM7:** wakeup with event generation mask on event input 7

Bit 6 **EM6:** wakeup with event generation mask on event input 6

Bit 5 **EM5:** wakeup with event generation mask on event input 5

Bit 4 **EM4:** wakeup with event generation mask on event input 4

Bit 3 **EM3:** wakeup with event generation mask on event input 3

Bit 2 **EM2:** wakeup with event generation mask on event input 2

Bit 1 **EM1:** wakeup with event generation mask on event input 1

Bit 0 **EM0:** wakeup with event generation mask on event input 0

14.6.11 EXTI interrupt mask register (EXTI_IMR2)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	IM46	IM45	IM44	IM43	IM42	Res.	Res.	Res.	IM38	Res.	Res.	Res.	IM34	Res.	Res.
	rw	rw	rw	rw	rw				rw				rw		

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **IM46:** wakeup with interrupt mask on event input 46
0: Wakeup with interrupt request from line 46 is masked.
1: Wakeup with interrupt request from line 46 is unmasked.

Bit 13 **IM45:** wakeup with interrupt mask on event input 45

Bit 12 **IM44:** wakeup with interrupt mask on event input 44

Bit 11 **IM43:** wakeup with interrupt mask on event input 43

Bit 10 **IM42:** wakeup with interrupt mask on event input 42

Bits 9:7 Reserved, must be kept at reset value.

Bit 6 **IM38:** wakeup with interrupt mask on event input 38

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **IM34:** wakeup with interrupt mask on event input 34

Bits 1:0 Reserved, must be kept at reset value.

14.6.12 EXTI register map

Table 85. EXTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	Res.	Res.	Res.	Res.	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value										0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	Res.	Res.	Res.	Res.	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value										0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI22	SWI21	Res.	Res.	Res.	Res.	SWI16	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset value										0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 85. EXTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	EXTI_PR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF22	PIF21	Res.	Res.	Res.	Res.	PIF16	PIF15	PIF14	PIF13	PIF12	PIF11	PIF10	PIF9	PIF8	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
	Reset value										0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010-0x01C	Reserved	Reserved.																															
0x020	EXTI_RTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT45	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT34	Res.	Res.
	Reset value																			0											0		
0x024	EXTI_FTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT45	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT34	Res.	Res.
	Reset value																			0											0		
0x028	EXTI_SWIER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI45	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI34	Res.	Res.
	Reset value																			0											0		
0x02C	EXTI_PR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF45	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF34	Res.	Res.
	Reset value																			0											0		
0x030-0x07C	Reserved	Reserved.																															
0x080	EXTI_IMR1	IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	EXTI_EMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM22	EM21	EM20	EM19	EM18	EM17	Res.	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value										0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	Reserved	Reserved.																															
0x090	EXTI_IMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM46	IM45	IM44	IM46	IM42	Res.	Res.	Res.	IM38	Res.	Res.	Res.	IM34	Res.	Res.
	Reset value																		0	0	0	0	0				0				0		

Refer to *Section 2.4* for the register boundary addresses.

15 Cyclic redundancy check calculation unit (CRC)

15.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

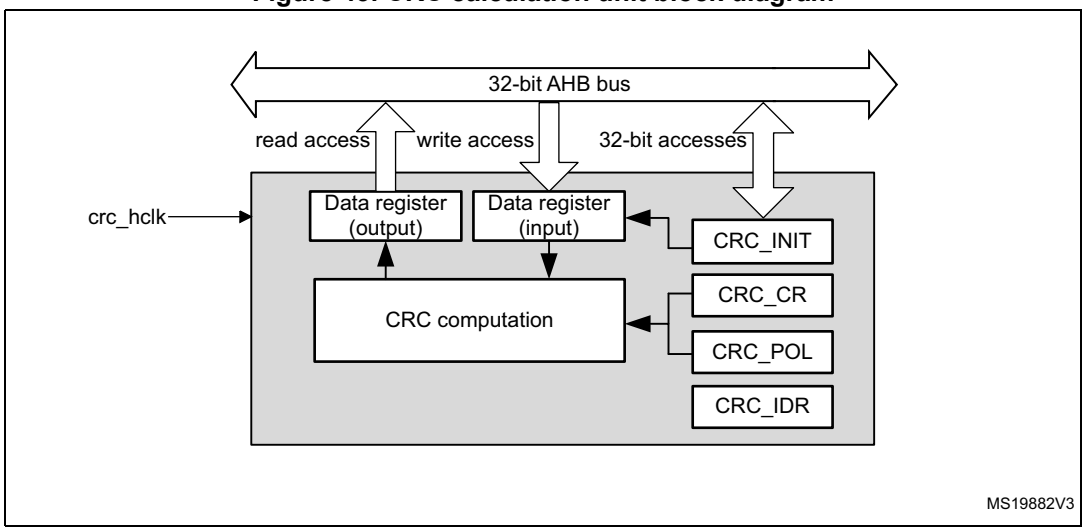
15.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

15.3 CRC functional description

15.3.1 CRC block diagram

Figure 43. CRC calculation unit block diagram



15.3.2 CRC internal signals

Table 86. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

15.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

15.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

15.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

15.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

15.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rs

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **REV_OUT**: Reverse output data

This bit controls the reversal of the bit order of the output data.

0: Bit order not affected

1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data

This bitfield controls the reversal of the bit order of the input data

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

15.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_INIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value
This register is used to write the CRC initial value.

15.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **POL[31:0]**: Programmable polynomial
This register is used to write the coefficients of the polynomial to be used for CRC calculation.
If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

15.4.6 CRC register map

Table 87. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**CRC_DR**	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	**CRC_IDR**	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**CRC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																									0	0	0	0	0			0
0x10	**CRC_INIT**	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	**CRC_POL**	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.4 on page 62* for the register boundary addresses.

16 Analog-to-digital converter (ADC)

16.1 Introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 18 multiplexed channels allowing it to measure signals from 12 external and 4 internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined higher or lower thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

A built-in hardware oversampler allows analog performances to be improved while off-loading the related computational burden from the CPU.

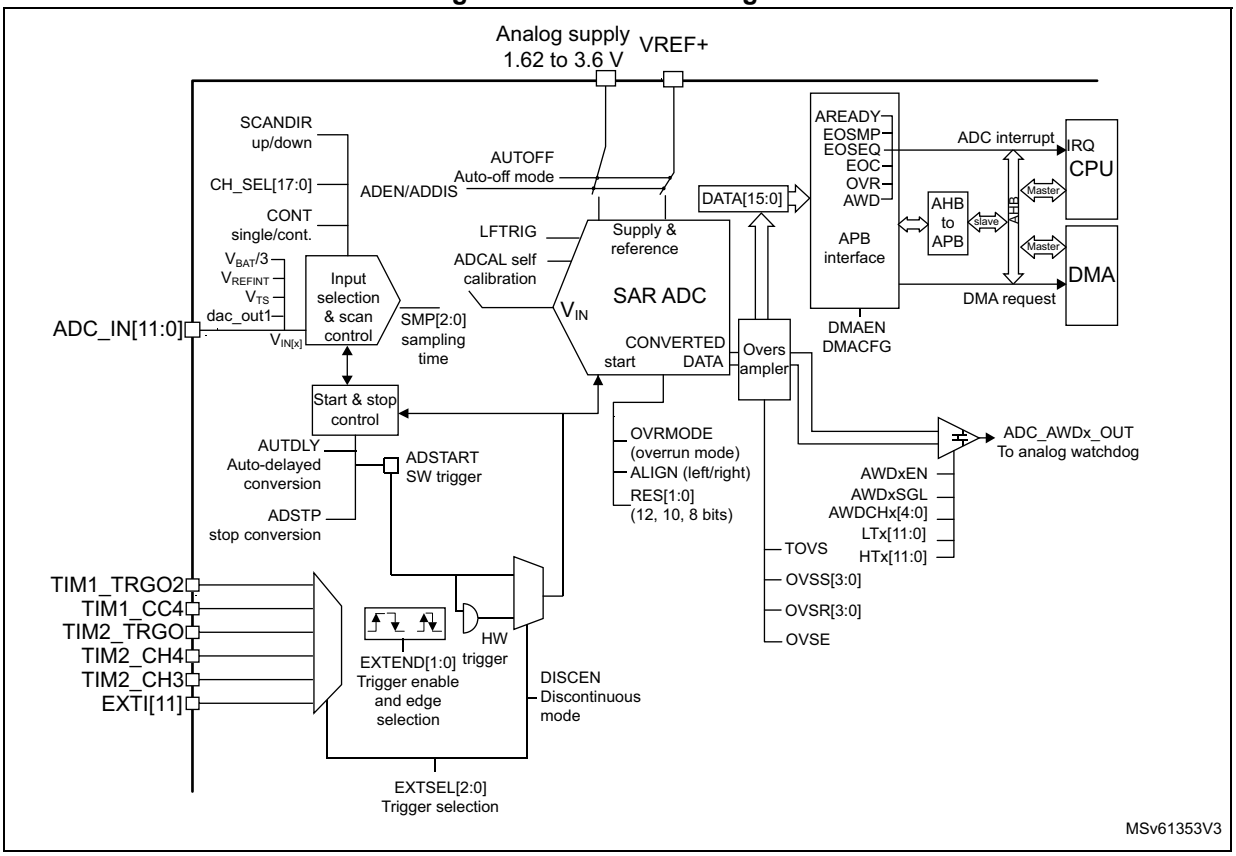
16.2 ADC main features

- High performance
  - 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
  - ADC conversion time: 0.4 µs for 12-bit resolution (2.5Msps), faster conversion times can be obtained by lowering resolution.
  - Self-calibration
  - Programmable sampling time
  - Data alignment with built-in data coherency
  - DMA support
- Low-power
  - The application can reduce PCLK frequency for low-power operation while still keeping optimum ADC performance. For example, 0.4 µs conversion time is kept, whatever the PCLK frequency
  - Wait mode: prevents ADC overrun in applications with low PCLK frequency
  - Auto off mode: ADC is automatically powered off except during the active conversion phase. This dramatically reduces the power consumption of the ADC.
- Analog input channels
  - 12 external analog inputs
  - 1 channel for internal temperature sensor (V_{TS})
  - 1 channel for internal reference voltage (V_{REFINT})
  - 1 channel for monitoring external V_{BAT} power supply pin
  - 1 channel for monitoring DAC internal channel input
- Start-of-conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (timer events or GPIO input events)
- Conversion modes
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at the end of sampling, end of conversion, end of sequence conversion, and in case of analog watchdog or overrun events
- Analog watchdog
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8-bits
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

16.3 ADC functional description

Figure 44 shows the ADC block diagram and *Table 88* gives the ADC pin description.

Figure 44. ADC block diagram



16.3.1 ADC pins and internal signals

Table 88. ADC input/output pins

Name	Signal type	Remarks
VDDA	Input, analog power supply	Analog power supply and positive reference voltage for the ADC
VSSA	Input, analog supply ground	Ground for analog power supply
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
ADC_INx	Analog input signals	12 external analog input channels

Table 89. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{IN}[x]$	Analog Input channels	Connected either to internal channels or to ADC_IN*i* external channels
TRGx	Input	ADC conversion triggers
V_{TS}	Input	Internal temperature sensor output voltage
V_{REFINT}	Input	Internal voltage reference output voltage
$V_{BAT/3}$	Input	VBAT pin input voltage divided by 3
dac_out1	Input	DAC internal channel1 input
ADC_AWDx_OUT	Output	Internal analog watchdog output signal connected to on-chip timers (x = Analog watchdog number = 1,2,3)

Table 90. External triggers

Name	Source	EXTSEL[2:0]
TRG0	TIM1_TRGO2	000
TRG1	TIM1_CC4	001
TRG2	TIM2_TRGO	010
TRG3	TIM2_CH4	011
TRG4	Reserved	100
TRG5	TIM2_CH3	101
TRG6	Reserved	110
TRG7	EXTI11	111

16.3.2 ADC voltage regulator (ADVREGEN)

The ADC has a specific internal voltage regulator which must be enabled and stable before using the ADC.

The ADC internal voltage regulator can be enabled by setting ADVREGEN bit to 1 in the ADC_CR register. The software must wait for the ADC voltage regulator startup time ($t_{ADCVREG\_STUP}$) before launching a calibration or enabling the ADC. This delay must be managed by software (for details on $t_{ADCVREG\_STUP}$, refer to the device datasheet).

After ADC operations are complete, the ADC is disabled (ADEN = 0). To keep power consumption low, it is important to disable the ADC voltage regulator before entering low-power mode (LPRun, LPSleep or Stop mode). Refer to *Section : ADC voltage regulator disable sequence*).

Note: When the internal voltage regulator is disabled, the internal analog calibration is kept.

Analog reference from the power control unit

The internal ADC voltage regulator internally uses an analog reference delivered by the power control unit through a buffer. This buffer is always enabled when the main voltage

regulator of the power control unit operates in normal Run mode (refer to Reset and clock control and power control sections).

If the main voltage regulator enters low-power mode (such as Low-power run mode), this buffer is disabled and the ADC cannot be used.

ADC Voltage regulator enable sequence

To enable the ADC voltage regulator, set ADVREGEN bit to 1 in ADC_CR register.

ADC voltage regulator disable sequence

To disable the ADC voltage regulator, follow the sequence below:

1. Make sure that the ADC is disabled (ADEN = 0).
2. Clear ADVREGEN bit in ADC_CR register.

16.3.3 Calibration (ADCAL)

The ADC has a calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is complete.

Calibration should be performed before starting A/D conversion. It removes the offset error which may vary from chip to chip due to process variation.

The calibration is initiated by software by setting bit ADCAL to 1. It can be initiated only when all the following conditions are met:

- the ADC voltage regulator is enabled (ADVREGEN = 1 and LDORDY = 1),
- the ADC is disabled (ADEN = 0), and
- the Auto-off mode is disabled (AUTOFF = 0).

ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. After this, the calibration factor can be read from the ADC_DR register (from bits 6 to 0).

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). When the ADC operating conditions change (V_{DDA} changes are the main contributor to ADC offset variations and temperature change to a lesser extend), it is recommended to re-run a calibration cycle.

The calibration factor is lost in the following cases:

- The power supply is removed from the ADC (for example when the product enters Standby or VBAT mode)
- The ADC peripheral is reset.

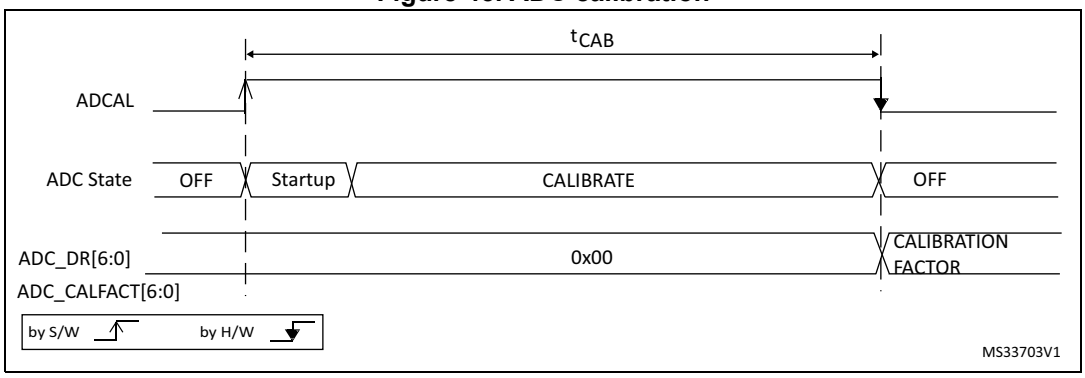
The calibration factor is lost each time power is removed from the ADC (for example when the product enters Standby or VBAT mode). Still, it is possible to save and restore the calibration factor by software to save time when re-starting the ADC (as long as temperature and voltage are stable during the ADC power-down).

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion.

Software calibration procedure

1. Ensure that ADEN = 0, AUTOFF = 0, ADVREGEN = 1 and DMAEN = 0.
2. Set ADCAL = 1.
3. Wait until ADCAL = 0 (or until EOCAL = 1). This can be handled by interrupt if the interrupt is enabled by setting the EOCALIE bit in the ADC_IER register
4. The calibration factor can be read from bits 6:0 of ADC_DR or ADC_CALFACT registers.
5. To reduce the noise effect of the calibration factor extraction, the software can make average of eight CALFACT[6:0] values (optional).

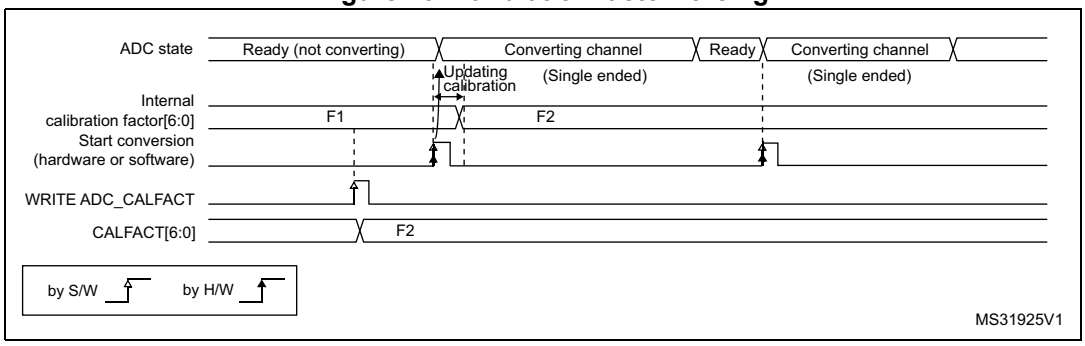
Figure 45. ADC calibration



Calibration factor forcing Software Procedure

1. Ensure that ADEN = 1 and ADSTART = 0 (ADC started with no conversion ongoing)
2. Write ADC_CALFACT with the saved calibration factor
3. The calibration factor is used as soon as a new conversion is launched.

Figure 46. Calibration factor forcing



16.3.4 ADC on-off control (ADEN, ADDIS, ADRDY)

At power-up, the ADC is disabled and put in power-down mode (ADEN = 0).

As shown in *Figure 47*, the ADC needs a stabilization time of t_{STAB} before it starts converting accurately.

Two control bits are used to enable or disable the ADC:

- Set ADEN = 1 to enable the ADC. The ADRDY flag is set as soon as the ADC is ready for operation.
- Set ADDIS = 1 to disable the ADC and put the ADC in power down mode. The ADEN and ADDIS bits are then automatically cleared by hardware as soon as the ADC is fully disabled.

Conversion can then start either by setting ADSTART to 1 (refer to *Section 16.4: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN) on page 442*) or when an external trigger event occurs if triggers are enabled.

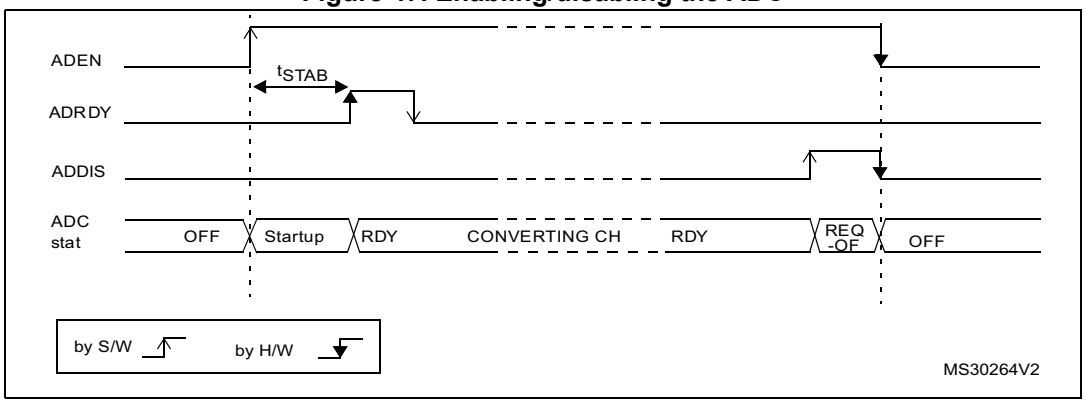
Follow this procedure to enable the ADC:

1. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1.
2. Set ADEN = 1 in the ADC_CR register.
3. Wait until ADRDY = 1 in the ADC_ISR register (ADRDY is set after the ADC startup time). This can be handled by interrupt if the interrupt is enabled by setting the ADRDYIE bit in the ADC_IER register.

Follow this procedure to disable the ADC:

1. Check that ADSTART = 0 in the ADC_CR register to ensure that no conversion is ongoing. If required, stop any ongoing conversion by writing 1 to the ADSTP bit in the ADC_CR register and waiting until this bit is read at 0.
2. Set ADDIS = 1 in the ADC_CR register.
3. If required by the application, wait until ADEN = 0 in the ADC_CR register, indicating that the ADC is fully disabled (ADDIS is automatically reset once ADEN = 0).
4. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1 (optional).

Figure 47. Enabling/disabling the ADC

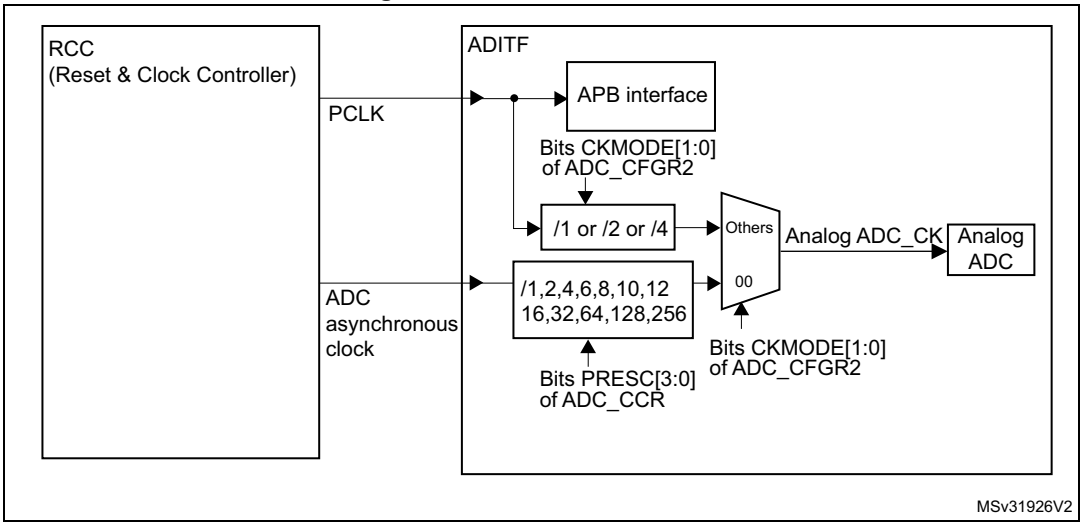


Note: In Auto-off mode (AUTOFF = 1) the power-on/off phases are performed automatically, by hardware and the ADRDY flag is not set.

16.3.5 ADC clock (CKMODE, PRESC[3:0])

The ADC has a dual clock-domain architecture, so that the ADC can be fed with a clock (ADC asynchronous clock) independent from the APB clock (PCLK).

Figure 48. ADC clock scheme



1. Refer to *Section Reset and clock control (RCC)* for how the PCLK clock and ADC asynchronous clock are enabled.

The input clock of the analog ADC can be selected between two different clock sources (see *Figure 48: ADC clock scheme* to see how the PCLK clock and the ADC asynchronous clock are enabled):

a) The ADC clock can be a specific clock source, named “ADC asynchronous clock“ which is independent and asynchronous with the APB clock.

Refer to RCC Section for more information on generating this clock source.

To select this scheme, bits CKMODE[1:0] of the ADC_CFGR2 register must be reset.

b) The ADC clock can be derived from the APB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4) according to bits CKMODE[1:0].

To select this scheme, bits CKMODE[1:0] of the ADC_CFGR2 register must be different from “00”.

In option a), the generated ADC clock can eventually be divided by a prescaler (1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128, 256) when programming the bits PRESC[3:0] in the ADC_CCR register).

Option a) has the advantage of reaching the maximum ADC clock frequency whatever the APB clock scheme selected.

Option b) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

Table 91. Latency between trigger and start of conversion(1)

ADC clock source	CKMODE[1:0]	Latency between the trigger event and the start of conversion
HSI16, SYSCLK or PLLPCLK(2)	00	Latency is not deterministic (jitter)
PCLK divided by 2	01	Latency is deterministic (no jitter) and equal to 3.25 ADC clock cycles
PCLK divided by 4	10	Latency is deterministic (no jitter) and equal to 3.125 ADC clock cycles
PCLK divided by 1	11	Latency is deterministic (no jitter) and equal to 3 ADC clock cycles

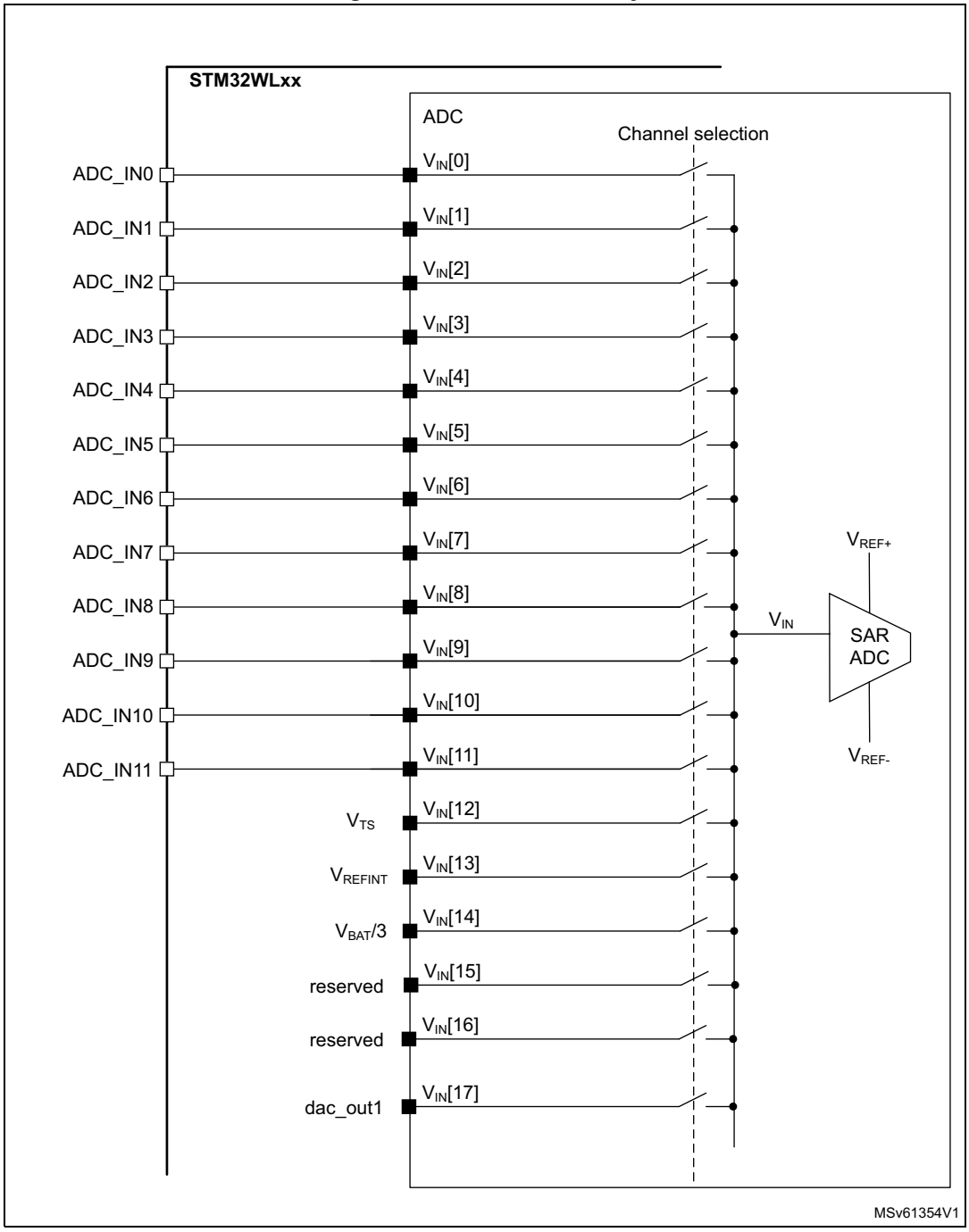
1. Refer to the device datasheet for the maximum ADC_CLK frequency.
2. Selected with ADCSEL bitfield of the RCC_CCIPR register

Caution: When selecting CKMODE[1:0] = 11 (PCLK divided by 1), the user must ensure that the PCLK has a 50% duty cycle. This is done by selecting a system clock with a 50% duty cycle and configuring the APB prescaler in bypass modes in the RCC (refer to there Reset and clock controller section). If an internal source clock is selected, the AHB and APB prescalers do not divide the clock.

16.3.6 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described in *Figure 49*.

Figure 49. ADC connectivity



16.3.7 Configuring the ADC

The software must write the ADCAL and ADEN bits in the ADC_CR register and configure the ADC_CFGR1 and ADC_CFGR2 registers only when the ADC is disabled (ADEN must be cleared).

The software must only write to the ADSTART and ADDIS bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN = 1 and ADDIS = 0).

For all the other control bits in the ADC_IER, ADC_SMPR, ADC_CHSELR and ADC_CCR registers, refer to the description of the corresponding control bit in *Section 16.12: ADC registers*.

ADC_AWDTRx registers can be modified when conversion is ongoing.

The software must only write to the ADSTP bit in the ADC_CR register if the ADC is enabled (and possibly converting) and there is no pending request to disable the ADC (ADSTART = 1 and ADDIS = 0).

Note: There is no hardware protection preventing software from making write operations forbidden by the above rules. If such a forbidden write access occurs, the ADC may enter an undefined state. To recover correct operation in this case, the ADC must be disabled (clear ADEN = 0 and all the bits in the ADC_CR register).

16.3.8 Channel selection (CHSEL, SCANDIR, CHSELRMOD)

There are up to 18 multiplexed channels:

- 12 analog inputs from GPIO pins (ADC_INx)
- 4 internal analog inputs (Temperature Sensor, Internal Reference Voltage, DAC internal channel1, V_{BAT} channel)
- 2 reserved analog inputs (ADC V_{IN}[16] and V_{IN}[15])

It is possible to convert a single channel or a sequence of channels.

The sequence of the channels to be converted can be programmed in the ADC_CHSELR channel selection register: each analog input channel has a dedicated selection bit (CHSELx).

The ADC scan sequencer can be used in two different modes:

- Sequencer not fully configurable:

  The order in which the channels are scanned is defined by the channel number (CHSELRMOD bit must be cleared in ADC_CFGR1 register):

  – Sequence length configured through CHSELx bits in ADC_CHSELR register

  – Sequence direction: the channels are scanned in a forward direction (from the lowest to the highest channel number) or backward direction (from the highest to the lowest channel number) depending on the value of SCANDIR bit (SCANDIR = 0: forward scan, SCANDIR = 1: backward scan)

  - Any channel can belong to in these sequences
- Sequencer fully configurable

  The CHSELRMOD bit is set in ADC_CFGR1 register.
  - Sequencer length is up to 8 channels
  - The order in which the channels are scanned is independent from the channel number. Any order can be configured through SQ1[3:0] to SQ8[3:0] bits in ADC_CHSELR register.
  - Only channel 0 to channel 14 can be selected in this sequence
  - If the sequencer detects SQx[3:0] = 0b1111, the following SQx[3:0] registers are ignored.
  - If no 0b1111 is programmed in SQx[3:0], the sequencer scans full eight channels.

After programming ADC CHSELR, SCANDIR and CHSELRMOD bits, it is mandatory to wait for CCRDY flag before starting conversions. It indicates that the new channel setting has been applied. If a new configuration is required, the CCRDY flag must be cleared prior to starting the conversion.

The software is allowed to program the CHSEL, SCANDIR, CHSELRMOD bits only when ADSTART bit is cleared (which ensures that no conversion is ongoing).

Temperature sensor, DAC output, V_{REFINT} and V_{BAT} internal channels

The temperature sensor is connected to channel ADC V_{IN}[12].

The internal voltage reference V_{REFINT} is connected to channel ADC V_{IN}[13].

V_{BAT} channel is connected to ADC V_{IN}[14] channel.

The internal dac_out1 output voltage is connected to ADC V_{IN}[17] channel.

16.3.9 Programmable sampling time (SMPx[2:0])

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the sample and hold capacitor to the input voltage level.

Having a programmable sampling time allows the conversion speed to be trimmed according to the input resistance of the input voltage source.

The ADC samples the input voltage for a number of ADC clock cycles that can be modified using the SMP1[2:0] and SMP2[2:0] bits in the ADC_SMPR register.

Each channel can choose one out of two sampling times configured in SMP1[2:0] and SMP2[2:0] bitfields, through SMPSELx bits in ADC_SMPR register.

The total conversion time is calculated as follows:

t_{CONV} = Sampling time + 12.5 x ADC clock cycles

Example:

With ADC_CLK = 16 MHz and a sampling time of 1.5 ADC clock cycles:

t_{CONV} = 1.5 + 12.5 = 14 ADC clock cycles = 0.875 µs

The ADC indicates the end of the sampling phase by setting the EOSMP flag.

16.3.10 Single conversion mode (CONT = 0)

In Single conversion mode, the ADC performs a single sequence of conversions, converting all the channels once. This mode is selected when CONT = 0 in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

16.3.11 Continuous conversion mode (CONT = 1)

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting all the channels once and then automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when CONT = 1 in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

16.3.12 Starting conversions (ADSTART)

Software starts ADC conversions by setting ADSTART = 1.

When ADSTART is set, the conversion:

- Starts immediately if EXTEN = 00 (software trigger)
- At the next active edge of the selected hardware trigger if EXTEN ≠ 00

The ADSTART bit is also used to indicate whether an ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0, indicating that the ADC is idle.

The ADSTART bit is cleared by hardware:

- In single mode with software trigger (CONT = 0, EXTEN = 00)
  - At any end of conversion sequence (EOS = 1)
- In discontinuous mode with software trigger (CONT = 0, DISCEN = 1, EXTEN = 00)
  - At end of conversion (EOC = 1)
- In all cases (CONT = x, EXTEN = XX)
  - After execution of the ADSTP procedure invoked by software (see *Section 16.3.14: Stopping an ongoing conversion (ADSTP) on page 442*)

Note: In continuous mode (CONT = 1), the ADSTART bit is not cleared by hardware when the EOS flag is set because the sequence is automatically relaunched.

When hardware trigger is selected in single mode (CONT = 0 and EXTEN = 01), ADSTART is not cleared by hardware when the EOS flag is set (except if DMAEN = 1 and DMACFG = 0 in which case ADSTART is cleared at end of the DMA transfer). This avoids the need for software having to set the ADSTART bit again and ensures the next trigger event is not missed.

After changing channel selection configuration (by programming ADC_CHSELR register or changing CHSELRMOD or SCANDIR), it is mandatory to wait until CCRDY flag is asserted before asserting ADSTART, otherwise the value written to ADSTART is ignored.

16.3.13 Timings

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$t_{CONV} = t_{SMPL} + t_{SAR} = [1.5\ _{|min} + 12.5\ _{|12bit}] \times t_{ADC\_CLK}$

$t_{CONV} = t_{SMPL} + t_{SAR} = 42.9\ ns\ _{|min} + 357.1\ ns\ _{|12bit} = 0.400\ \mu s\ _{|min}$ (for $f_{ADC\_CLK} = 35$ MHz)

Figure 50. Analog to digital conversion time

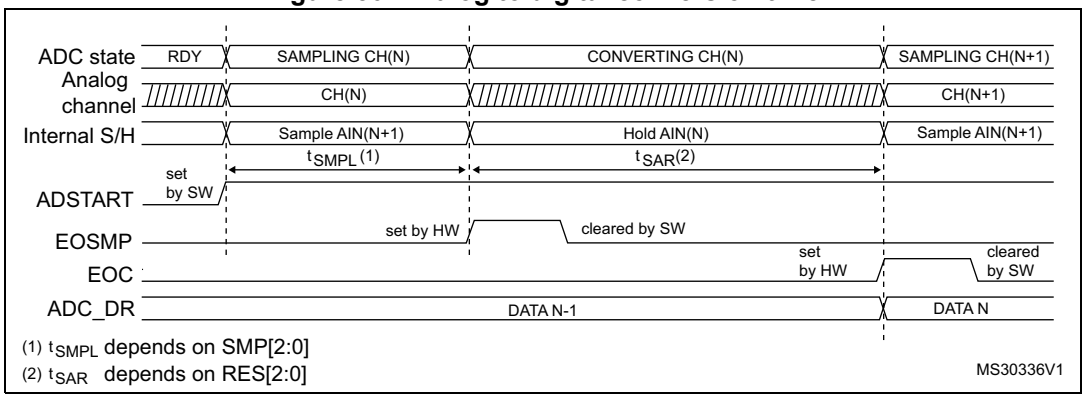
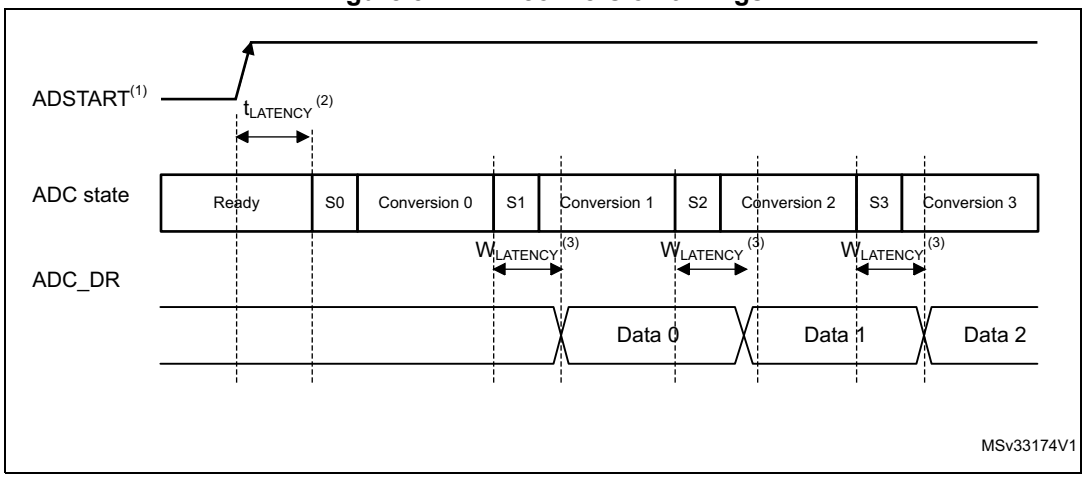


Figure 51. ADC conversion timings



1. EXTEN = 00 or EXTEN ≠ 00
2. Trigger latency (refer to datasheet for more details)
3. ADC_DR register write latency (refer to datasheet for more details)

16.3.14 Stopping an ongoing conversion (ADSTP)

The software can decide to stop any ongoing conversions by setting ADSTP = 1 in the ADC_CR register.

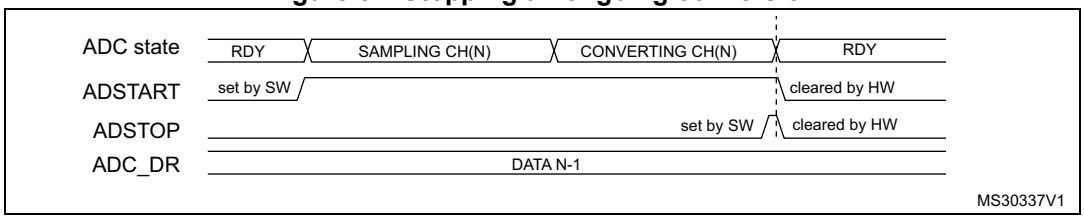
This resets the ADC operation and the ADC is idle, ready for a new operation.

When the ADSTP bit is set by software, any ongoing conversion is aborted and the result is discarded (ADC_DR register is not updated with the current conversion).

The scan sequence is also aborted and reset (meaning that restarting the ADC would re-start a new sequence).

Once this procedure is complete, the ADSTP and ADSTART bits are both cleared by hardware and the software must wait until ADSTART=0 before starting new conversions.

Figure 52. Stopping an ongoing conversion



16.4 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)

A conversion or a sequence of conversion can be triggered either by software or by an external event (for example timer capture). If the EXTEN[1:0] control bits are not equal to “0b00”, then external events are able to trigger a conversion with the selected polarity. The trigger selection is effective once software has set bit ADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

If bit ADSTART = 0, any hardware triggers which occur are ignored.

Table 92 provides the correspondence between the EXTEN[1:0] values and the trigger polarity.

Table 92. Configuring the trigger polarity

Source	EXTEN[1:0]
Trigger detection disabled	00
Detection on rising edge	01
Detection on falling edge	10
Detection on both rising and falling edges	11

Note: The polarity of the external trigger can be changed only when the ADC is not converting (ADSTART = 0).

The EXTSEL[2:0] control bits are used to select which of 8 possible events can trigger conversions.

Refer to *Table 90: External triggers* in *Section 16.3.1: ADC pins and internal signals* for the list of all the external triggers that can be used for regular conversion.

The software source trigger events can be generated by setting the ADSTART bit in the ADC_CR register.

Note: The trigger selection can be changed only when the ADC is not converting (ADSTART = 0).

16.4.1 Discontinuous mode (DISCEN)

This mode is enabled by setting the DISCEN bit in the ADC_CFGR1 register.

In this mode (DISCEN = 1), a hardware or software trigger event is required to start each conversion defined in the sequence. On the contrary, if DISCEN = 0, a single hardware or software trigger event successively starts all the conversions defined in the sequence.

Example:

- DISCEN = 1, channels to be converted = 0, 3, 7, 10
  - 1st trigger: channel 0 is converted and an EOC event is generated
  - 2nd trigger: channel 3 is converted and an EOC event is generated
  - 3rd trigger: channel 7 is converted and an EOC event is generated
  - 4th trigger: channel 10 is converted and both EOC and EOS events are generated.
  - 5th trigger: channel 0 is converted an EOC event is generated
  - 6th trigger: channel 3 is converted and an EOC event is generated
  - ...
- DISCEN = 0, channels to be converted = 0, 3, 7, 10
  - 1st trigger: the complete sequence is converted: channel 0, then 3, 7 and 10. Each conversion generates an EOC event and the last one also generates an EOS event.
  - Any subsequent trigger events restarts the complete sequence.

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

16.4.2 Programmable resolution (RES) - Fast conversion mode

It is possible to obtain faster conversion times (t_{SAR}) by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] bits in the ADC_CFGR1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

Note: The RES[1:0] bit must only be changed when the ADEN bit is reset.

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeros.

Lower resolution reduces the conversion time needed for the successive approximation steps as shown in *Table 93*.

Table 93. t_{SAR} timings depending on resolution

RES[1:0] bits	t_{SAR} (ADC clock cycles)	t_{SAR} (ns) at f_{ADC} = 35 MHz	$t_{SMPL\ (min)}$ (ADC clock cycles)	t_{CONV} (ADC clock cycles) (with min. t_{SMPL})	t_{CONV} (ns) at f_{ADC} = 35 MHz
12	12.5	357	1.5	14	400
10	10.5	300	1.5	12	343
8	8.5	243	1.5	10	286
6	6.5	186	1.5	8	229

16.4.3 End of conversion, end of sampling phase (EOC, EOSMP flags)

The ADC indicates each end of conversion (EOC) event.

The ADC sets the EOC flag in the ADC_ISR register as soon as a new conversion data result is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set in the ADC_IER register. The EOC flag is cleared by software either by writing 1 to it, or by reading the ADC_DR register.

The ADC also indicates the end of sampling phase by setting the EOSMP flag in the ADC_ISR register. The EOSMP flag is cleared by software by writing1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC_IER register.

The aim of this interrupt is to allow the processing to be synchronized with the conversions. Typically, an analog multiplexer can be accessed in hidden time during the conversion phase, so that the multiplexer is positioned when the next sampling starts.

Note: As there is only a very short time left between the end of the sampling and the end of the conversion, it is recommenced to use polling or a WFE instruction rather than an interrupt and a WFI instruction.

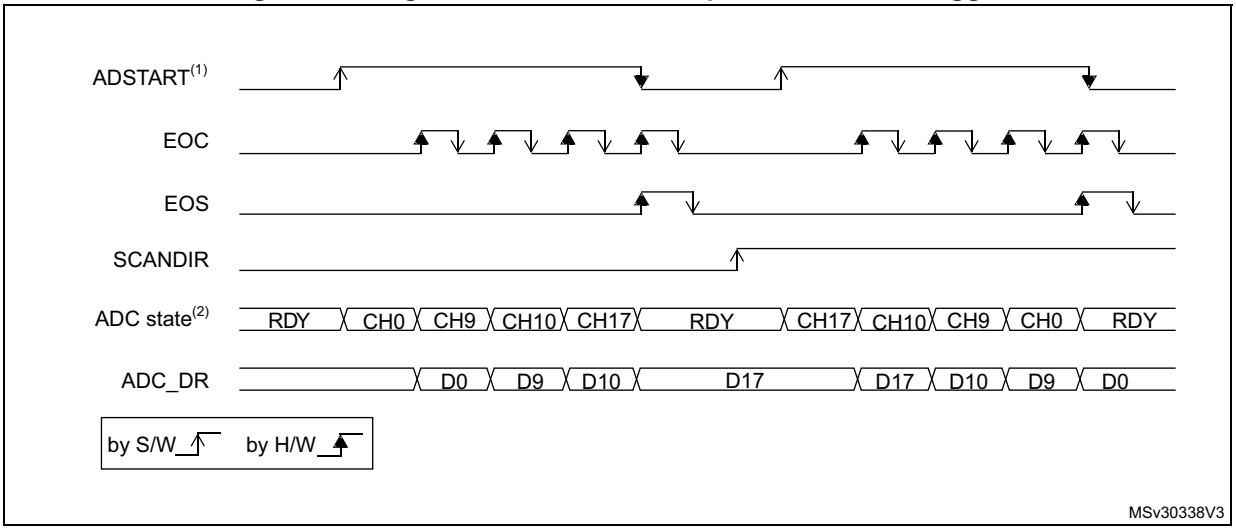
16.4.4 End of conversion sequence (EOS flag)

The ADC notifies the application of each end of sequence (EOS) event.

The ADC sets the EOS flag in the ADC_ISR register as soon as the last data result of a conversion sequence is available in the ADC_DR register. An interrupt can be generated if the EOSIE bit is set in the ADC_IER register. The EOS flag is cleared by software by writing 1 to it.

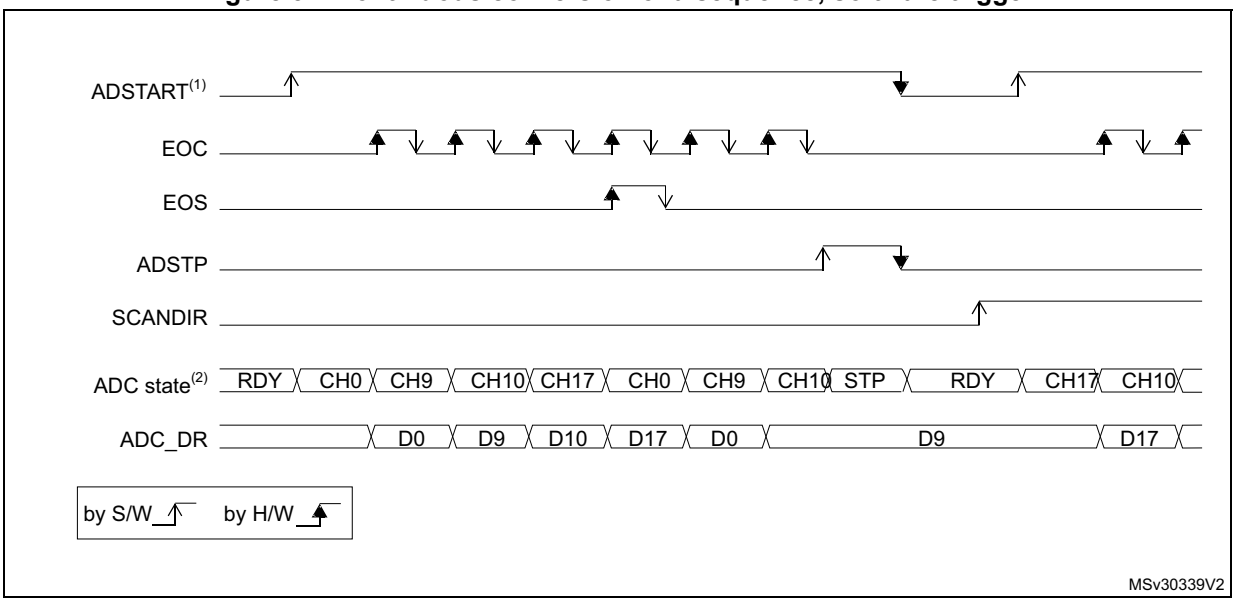
16.4.5 Example timing diagrams (single/continuous modes hardware/software triggers)

Figure 53. Single conversions of a sequence, software trigger



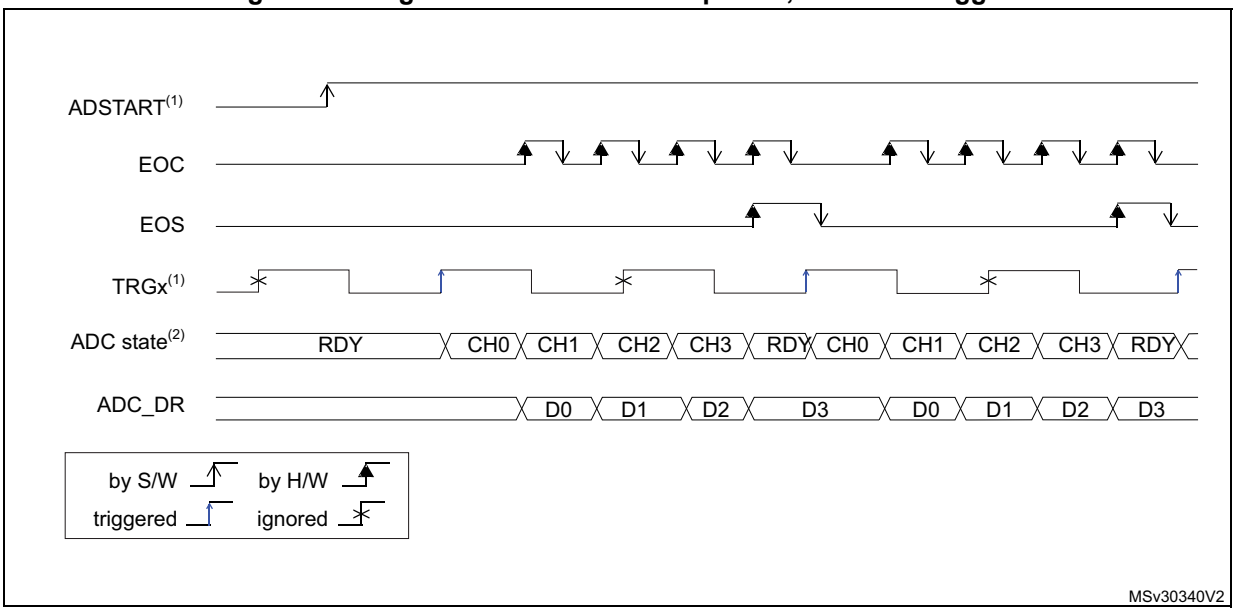
1. EXTEN = 00, CONT = 0
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

Figure 54. Continuous conversion of a sequence, software trigger



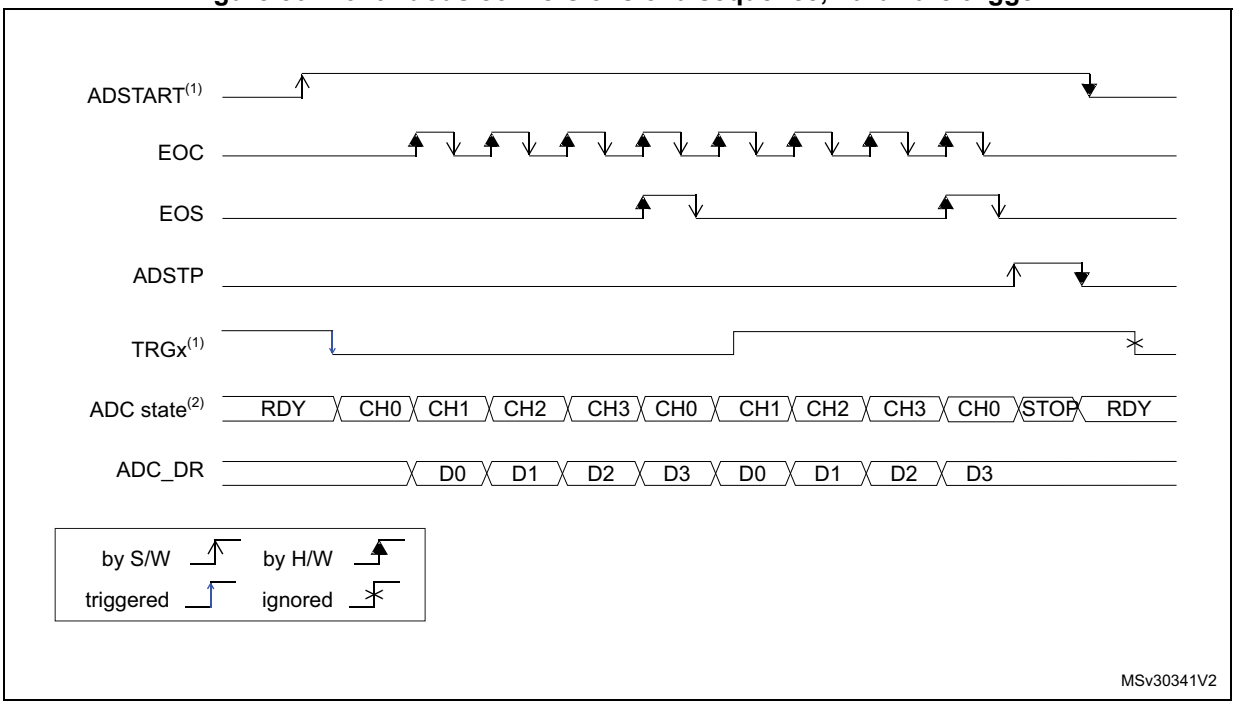
1. EXTEN = 00, CONT = 1,
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

Figure 55. Single conversions of a sequence, hardware trigger



1. EXTSEL = TRGx (over-frequency), EXTEN = 01 (rising edge), CONT = 0
2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0

Figure 56. Continuous conversions of a sequence, hardware trigger



1. EXTSEL = TRGx, EXTEN = 10 (falling edge), CONT = 1
2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0

16.4.6 Low frequency trigger mode

Once the ADC is enabled or the last ADC conversion is complete, the ADC is ready to start a new conversion. The ADC needs to be started at a predefined time (t_{idle}) otherwise ADC converted data might be corrupted due to the transistor leakage (refer to the device datasheet for the maximum value of t_{idle}).

If the application has to support a time longer than the maximum t_{idle} value (between one trigger to another for single conversion mode or between the ADC enable and the first ADC conversion), then the ADC internal state needs to be rearmed. This mechanism can be enabled by setting LFTRIG bit to 1 in ADC_CFGR2 register. By setting this bit, any trigger (software or hardware) sends a rearm command to ADC. The conversion starts after a one ADC clock cycle delay compared to LFTRIG cleared.

It is not necessary to use this mode when AUTOFF bit is set. For Wait mode, only the first trigger generates an internal rearm command.

16.5 Data management

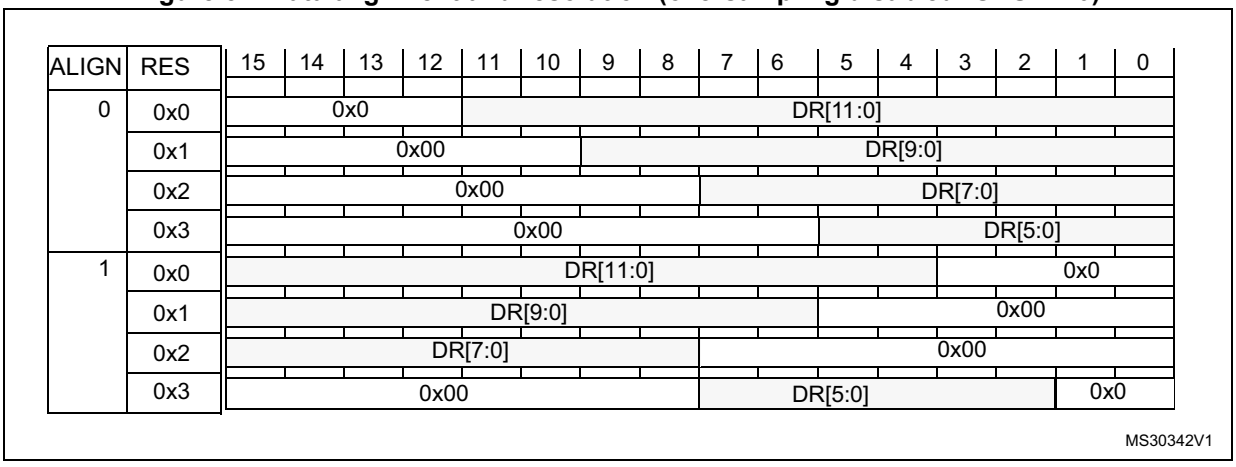
16.5.1 Data register and data alignment (ADC_DR, ALIGN)

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC_DR data register which is 16-bit wide.

The format of the ADC_DR depends on the configured data alignment and resolution.

The ALIGN bit in the ADC_CFGR1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN = 0) or left-aligned (ALIGN = 1) as shown in *Figure 57*.

Figure 57. Data alignment and resolution (oversampling disabled: OVSE = 0)



16.5.2 ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) indicates a data overrun event, when the converted data was not read in time by the CPU or the DMA, before the data from a new conversion is available.

The OVR flag is set in the ADC_ISR register if the EOC flag is still at ‘1’ at the time when a new conversion completes. An interrupt can be generated if the OVRIE bit is set in the ADC_IER register.

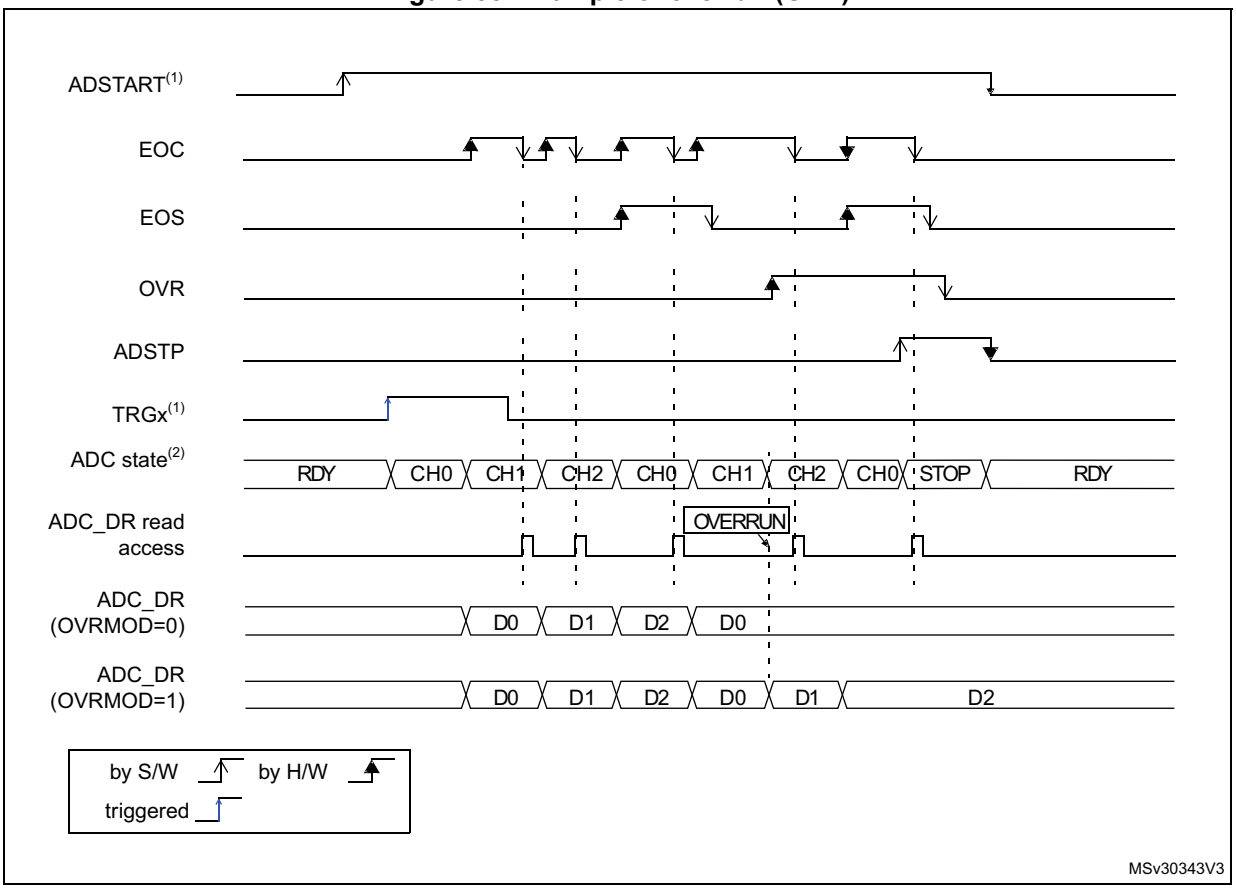
When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC_CR register.

The OVR flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or overwritten when an overrun event occurs by programming the OVRMOD bit in the ADC_CFGR1 register:

- OVRMOD = 0
  - An overrun event preserves the data register from being overwritten: the old data is maintained and the new conversion is discarded. If OVR remains at 1, further conversions can be performed but the resulting data is discarded.
- OVRMOD = 1
  - The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, further conversions can be performed and the ADC_DR register always contains the data from the latest conversion.

Figure 58. Example of overrun (OVR)



16.5.3 Managing a sequence of data converted without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software must use the EOC flag and its associated interrupt to handle each data result. Each time a conversion is complete, the EOC bit is set in the ADC_ISR register and the ADC_DR register can be read. The OVRMOD bit in the ADC_CFGR1 register should be configured to 0 to manage overrun events as an error.

16.5.4 Managing converted data without using the DMA without overrun

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the OVRMOD bit must be configured at 1 and the OVR flag should be ignored by the software. When OVRMOD = 1, an overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion data.

16.5.5 Managing converted data using the DMA

Since all converted channel values are stored in a single data register, it is efficient to use DMA when converting more than one channel. This avoids losing the conversion data results stored in the ADC_DR register.

When DMA mode is enabled (DMAEN bit set in the ADC_CFGR1 register), a DMA request is generated after the conversion of each channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Note: The DMAEN bit in the ADC_CFGR1 register must be set after the ADC calibration phase.

Despite this, if an overrun occurs (OVR = 1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to *Section 16.5.2: ADC overrun (OVR, OVRMOD) on page 447*).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG in the ADC_CFGR1 register:

- DMA one shot mode (DMACFG = 0).
  This mode should be selected when the DMA is programmed to transfer a fixed number of data words.
- DMA circular mode (DMACFG = 1)
  This mode should be selected when programming the DMA in circular mode or double buffer mode.

DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs, see *Section 10: Direct memory access controller (DMA) on page 190*) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted and its partial result discarded
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- The scan sequence is stopped and reset
- The DMA is stopped

DMA circular mode (DMACFG = 1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available in the data register, even if the DMA has reached the last DMA transfer. This allows the DMA to be configured in circular mode to handle a continuous analog input data stream.

16.6 Low-power features

16.6.1 Wait mode conversion

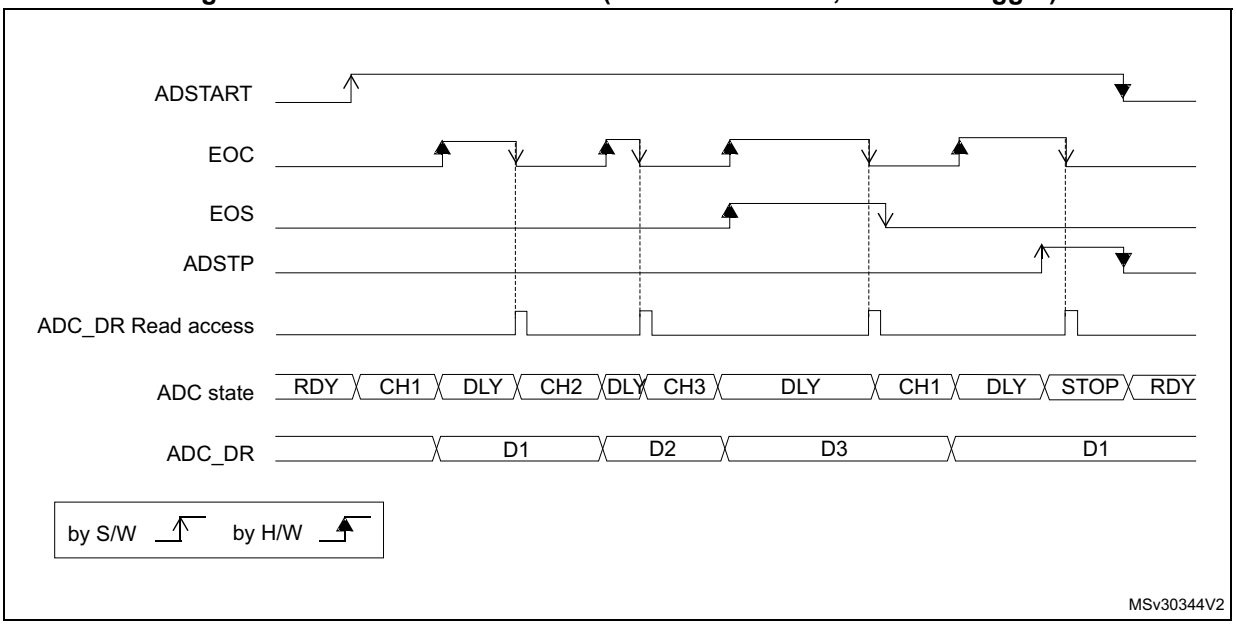
Wait mode conversion can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring.

When the WAIT bit is set in the ADC_CFGR1 register, a new conversion can start only if the previous data has been treated, once the ADC_DR register has been read or if the EOC bit has been cleared.

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data.

Note: Any hardware triggers which occur while a conversion is ongoing or during the wait time preceding the read access are ignored.

Figure 59. Wait mode conversion (continuous mode, software trigger)



1. EXTEN = 00, CONT = 1
2. CHSEL = 0x3, SCANDIR = 0, WAIT = 1, AUTOFF = 0

16.6.2 Auto-off mode (AUTOFF)

The ADC has an automatic power management feature which is called auto-off mode, and is enabled by setting AUTOFF = 1 in the ADC_CFGR1 register.

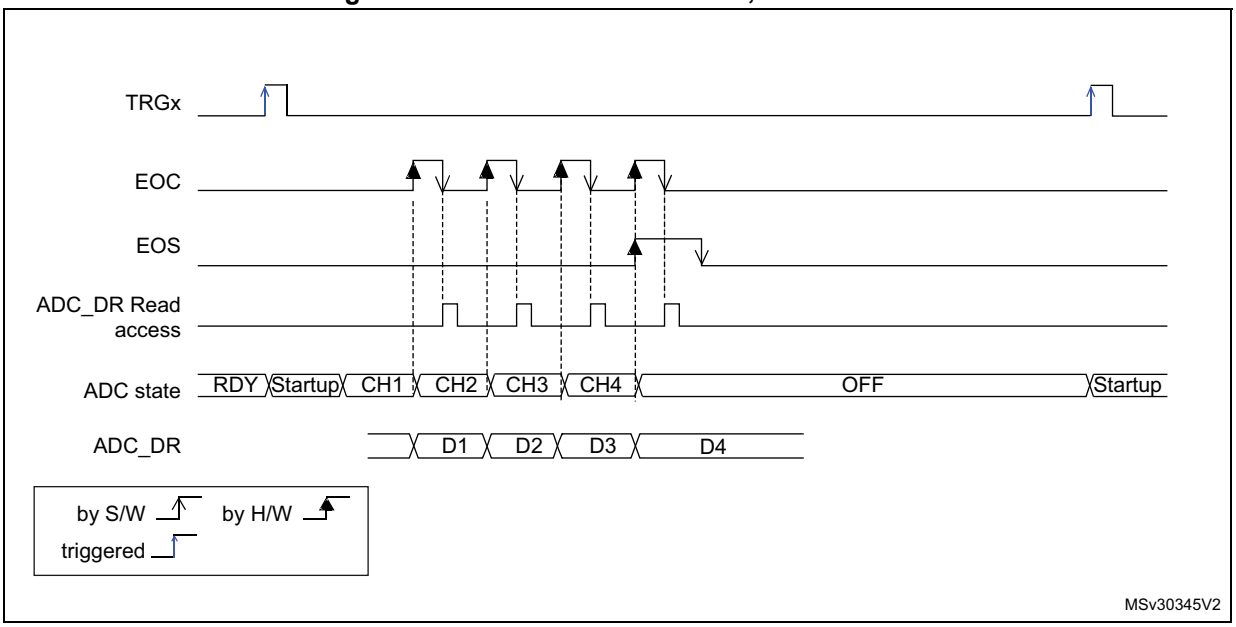
When AUTOFF = 1, the ADC is always powered off when not converting and automatically wakes-up when a conversion is started (by software or hardware trigger). A startup-time is automatically inserted between the trigger event which starts the conversion and the sampling time of the ADC. The ADC is then automatically disabled once the sequence of conversions is complete.

Auto-off mode can cause a dramatic reduction in the power consumption of applications which need relatively few conversions or when conversion requests are timed far enough apart (for example with a low frequency hardware trigger) to justify the extra power and extra time used for switching the ADC on and off.

Auto-off mode can be combined with the wait mode conversion (WAIT = 1) for applications clocked at low frequency. This combination can provide significant power savings if the ADC is automatically powered-off during the wait phase and restarted as soon as the ADC_DR register is read by the application (see *Figure 61: Behavior with WAIT = 1, AUTOFF = 1*).

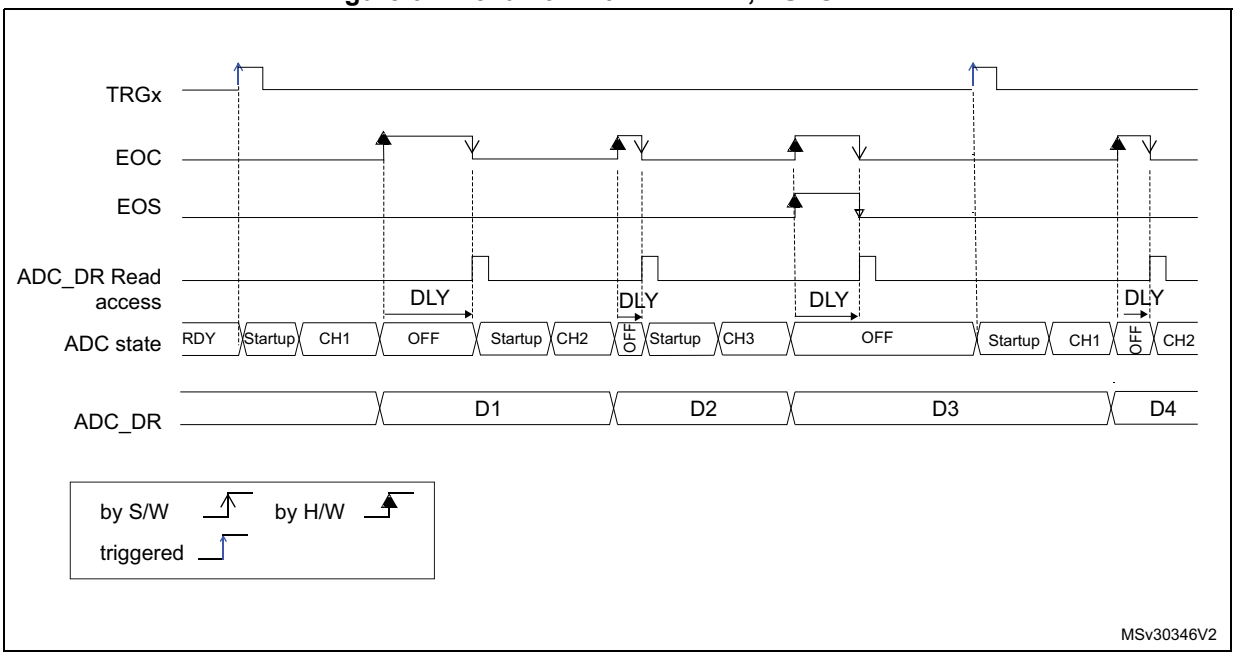
Note: Refer to the Section Reset and clock control (RCC) for the description of how to manage the dedicated 14 MHz internal oscillator. The ADC interface can automatically switch ON/OFF the 14 MHz internal oscillator to save power.

Figure 60. Behavior with WAIT = 0, AUTOFF = 1



1. EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 1, AUTOFF = 1

Figure 61. Behavior with WAIT = 1, AUTOFF = 1



1. EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 1, AUTOFF = 1

16.7 Analog window watchdogs

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

16.7.1 Description of analog watchdog 1

AWD1 analog watchdog is enabled by setting the AWD1EN bit in the ADC_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels (see *Table 95: Analog watchdog 1 channel selection*) remain within a configured voltage range (window) as shown in *Figure 62*.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in HT1[11:0] and LT1[11:0] bits of ADC_AWD1TR register. An interrupt can be enabled by setting the AWD1IE bit in the ADC_IER register.

The AWD1 flag is cleared by software by programing it to 1.

When converting data with a resolution of less than 12-bit (according to bits DRES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 94 describes how the comparison is performed for all the possible resolutions.

Table 94. Analog watchdog comparison

Resolution bits RES[1:0]	Analog Watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LTx[11:0] and HTx[11:0]	-
01: 10-bit	DATA[11:2],00	LTx[11:0] and HTx[11:0]	The user must configure LTx[1:0] and HTx[1:0] to “00”
10: 8-bit	DATA[11:4],0000	LTx[11:0] and HTx[11:0]	The user must configure LTx[3:0] and HTx[3:0] to “0000”
11: 6-bit	DATA[11:6],000000	LTx[11:0] and HTx[11:0]	The user must configure LTx[5:0] and HTx[5:0] to “000000”

1. The watchdog comparison is performed on the raw converted data before any alignment calculation.

Table 95 shows how to configure the AWD1SGL and AWD1EN bits in the ADC_CFGR1 register to enable the analog watchdog on one or more channels.

Figure 62. Analog watchdog guarded area

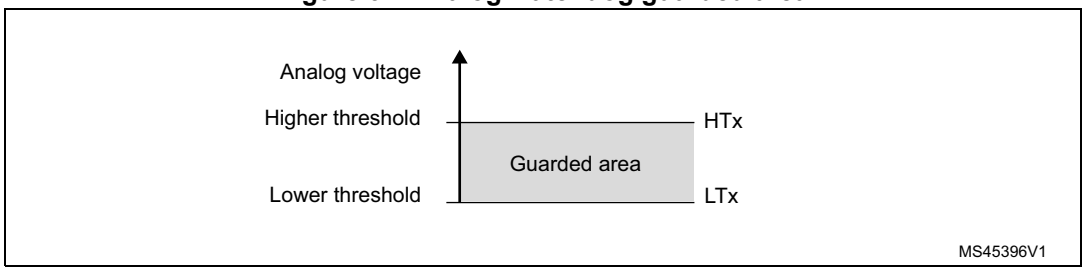


Table 95. Analog watchdog 1 channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit
None	x	0

Table 95. Analog watchdog 1 channel selection (continued)

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit
All channels	0	1
Single(1) channel	1	1

1. Selected by the AWD1CH[4:0] bits

16.7.2 Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the AWDxCHy in ADC_AWDxCR (x = 2, 3).

The corresponding watchdog is enabled when any AWDxCHy bit (x = 2,3) is set in ADC_AWDxCR register.

When converting data with a resolution of less than 12 bits (configured through DRES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 94 describes how the comparison is performed for all the possible resolutions.

The AWD2/3 analog watchdog status bit is set if the analog voltage converted by the ADC is below a low threshold or above a high threshold. These thresholds are programmed in HTx[11:0] and LTx[11:0] of ADC_AWDxTR registers (x = 2 or 3). An interrupt can be enabled by setting the AWDxIE bit in the ADC_IER register.

The AWD2 and ADW3 flags are cleared by software by programming them to 1.

16.7.3 ADC_AWDx_OUT output signal generation

Each analog watchdog is associated to an internal hardware signal, ADC_AWDx_OUT (x being the watchdog number) that is directly connected to the ETR input (external trigger) of some on-chip timers (refer to the timers section for details on how to select the ADC_AWDx_OUT signal as ETR).

ADC_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds. It remains at 1 if the next guarded conversions are still outside the programmed thresholds.
- ADC_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS to 1). Note that stopping conversions (ADSTP set), might clear the ADC_AWDx_OUT state.
- ADC_AWDx_OUT state does not change when the ADC converts the none-guarded channel (see *Figure 65*)

AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADC_AWDx_OUT (as an example, ADC_AWDx_OUT can toggle while AWDx flag remains at 1 if the software has not cleared the flag).

The ADC_AWDx_OUT signal is generated by the ADC_CLK domain. This signal can be generated even the APB clock is stopped.

The AWD comparison is performed at the end of each ADC conversion. The ADC_AWDx_OUT rising edge and falling edge occurs two ADC_CLK clock cycles after the comparison.

As ADC_AWDx_OUT is generated by the ADC_CLK domain and AWD flag is generated by the APB clock domain, the rising edges of these signals are not synchronized.

Figure 63. ADC_AWDx_OUT signal generation

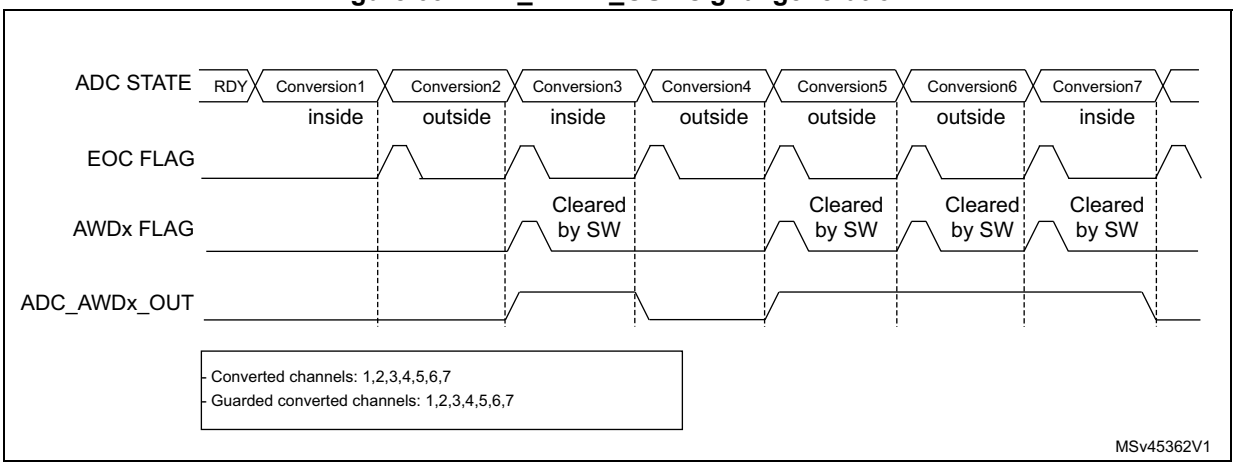


Figure 64. ADC_AWDx_OUT signal generation (AWDx flag not cleared by software)

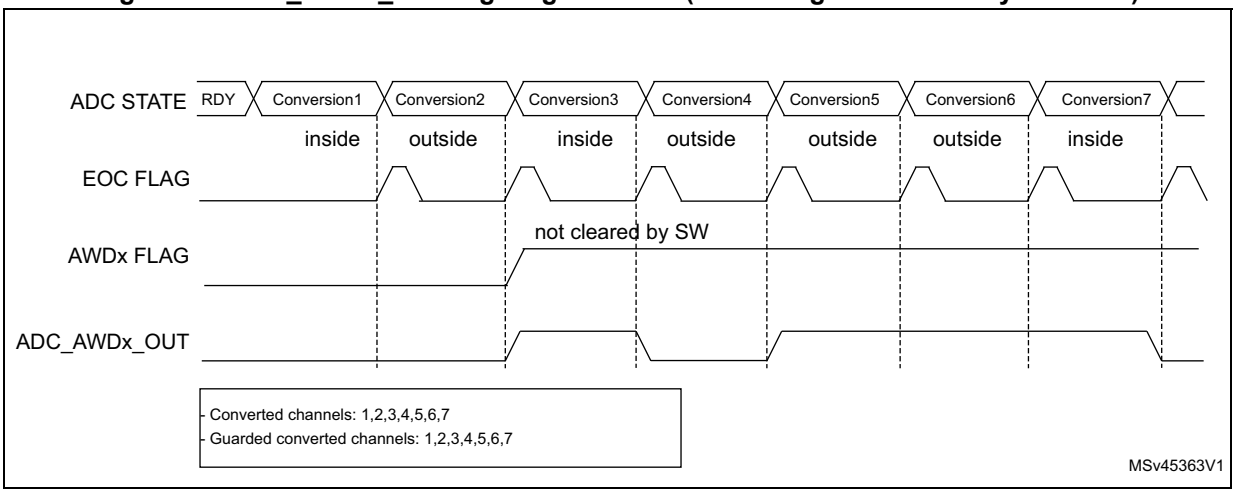
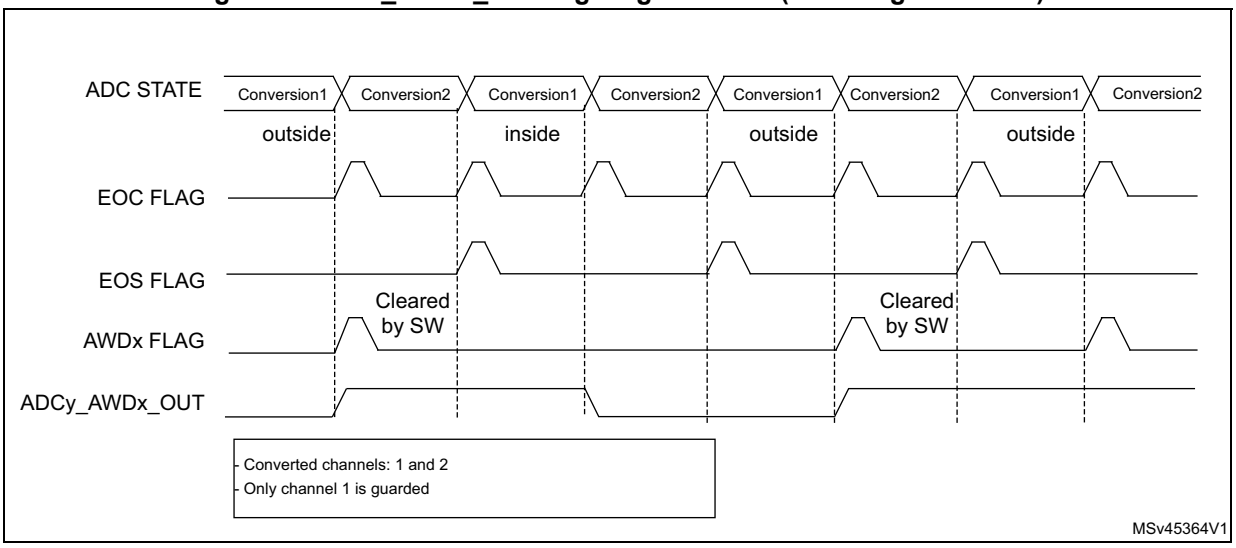


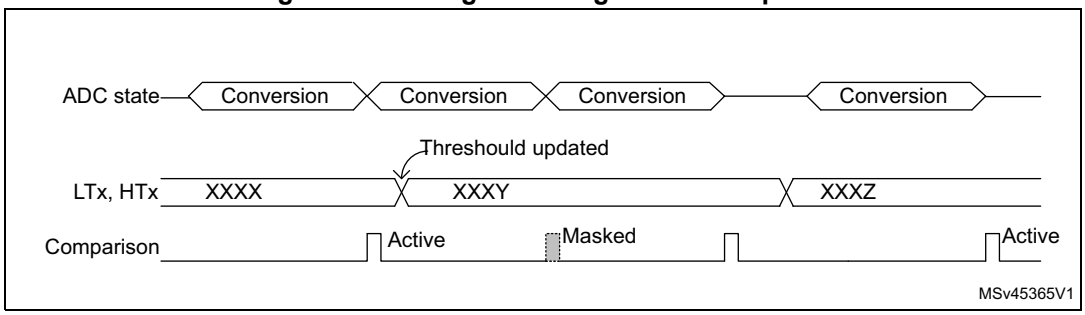
Figure 65. ADC_AWDx_OUT signal generation (on a single channel)



16.7.4 Analog Watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed during an analog-to-digital conversion (that is between the start of the conversion and the end of conversion of the ADC internal state). If HTx and LTx bits are programmed during the ADC guarded channel conversion, the watchdog function is masked for this conversion. This mask is cleared when starting a new conversion, and the resulting new AWD threshold is applied starting the next ADC conversion result. AWD comparison is performed at each end of conversion. If the current ADC data are out of the new threshold interval, this does not generated any interrupt or an ADC_AWDx_OUT signal. The Interrupt and the ADC_AWDx_OUT generation only occurs at the end of the ADC conversion that started after the threshold update. If ADC_AWDx_OUT is already asserted, programming the new threshold does not deassert the ADC_AWDx_OUT signal.

Figure 66. Analog watchdog threshold update



16.8 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

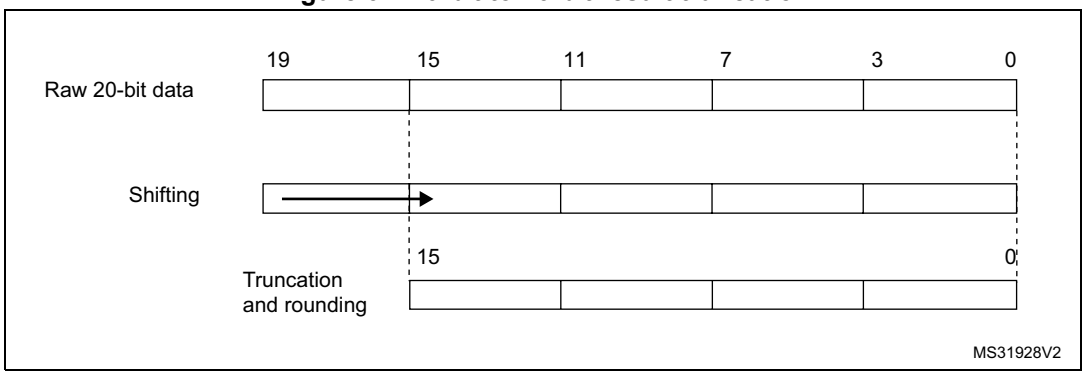
It allows the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC_CFGR2 register. It can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits. It is configured through the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256 x 12-bit), which is first shifted right. The upper bits of the result are then truncated, keeping only the 16 least significant bits rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

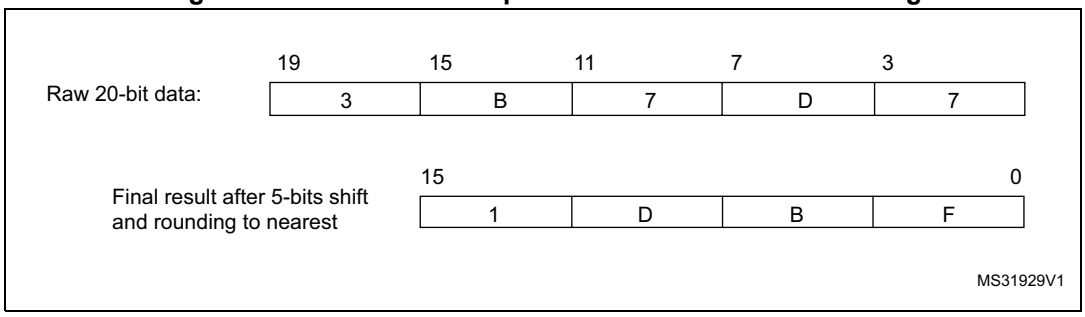
Note: If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

Figure 67. 20-bit to 16-bit result truncation



The *Figure 68* gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 68. Numerical example with 5-bits shift and rounding



The *Table 96* below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFF.

Table 96. Maximum output results vs N and M. Grayed values indicates truncation

Oversampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS = 0111	8-bit shift OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

The conversion timings in oversampled mode do not change compared to standard conversion mode: the sample time is maintained equal during the whole oversampling sequence. New data are provided every N conversion, with an equivalent delay equal to N x t_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags features are raised as following:

- the end of the sampling phase (EOSMP) is set after each sampling phase
- the end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- the end of sequence (EOCSEQ) occurs once the sequence of oversampled data is completed (i.e. after N x sequence length conversions total)

16.8.1 ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are available:

- Single or continuous mode conversions, forward or backward scanned sequences and up to 8 channels programmed sequence
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (WAIT, AUTOFF)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

Note: The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR1 is ignored and the data are always provided right-aligned.

16.8.2 Analog watchdog

The analog watchdog functionality is available, with the following differences:

- the RES[1:0] bits are ignored, comparison is always done on using the full 12-bits values HTx[11:0] and LTx[11:0]
- the comparison is performed on the most significant 12 bits of the 16 bits oversampled results ADC_DR[15:4]

Note: Care must be taken when using high shifting values. This reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits thus yielding a 12-bit data right-aligned, the affective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HTx[7:0] / LTx[[7:0], and HTx[11:8] / LTx[11:8] must be kept reset.

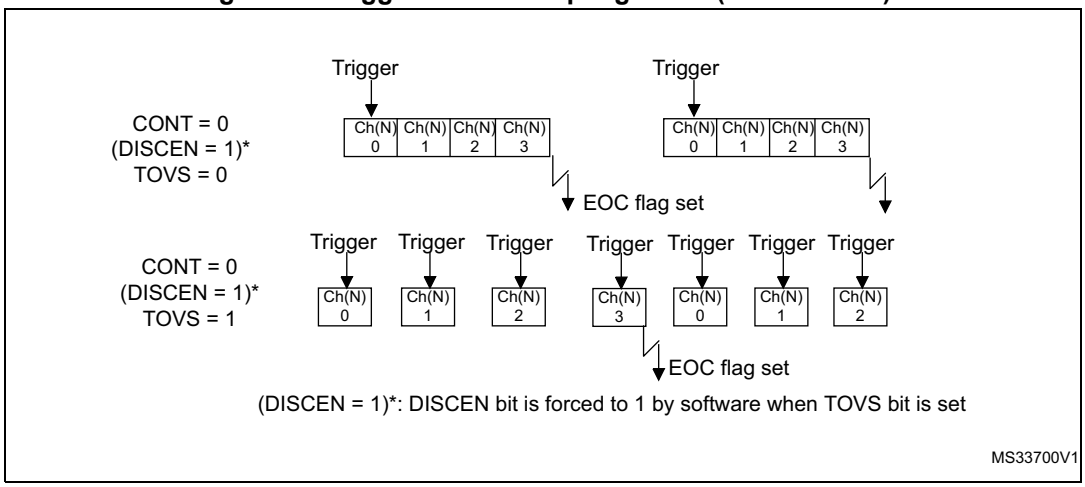
16.8.3 Triggered mode

The averager can also be used for basic filtering purposes. Although not a very efficient filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TOVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

Figure 69 below shows how conversions are started in response to triggers in discontinuous mode.

If the TOVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 69. Triggered oversampling mode (TOVS bit = 1)



16.9 Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the junction temperature (T_J) of the device. The temperature sensor is internally connected to the ADC V_{IN}[12] input channel which is used to convert the sensor's output voltage to a digital value. The sampling time for the temperature sensor analog pin must be greater than the minimum T_{S_temp} value specified in the datasheet. When not in use, the sensor can be put in power down mode.

The internal voltage reference (V_{REFINT}) provides a stable (bandgap) voltage output for the ADC and comparators. V_{REFINT} is internally connected to the ADC V_{IN}[13] input channel. The precise voltage of V_{REFINT} is individually measured for each part by ST during production test and stored in the system memory area.

Figure 70 shows the block diagram of connections between the temperature sensor, the internal voltage reference and the ADC.

The TSEN bit must be set to enable the conversion of ADC V_{IN}[12] (temperature sensor) and the VREFEN bit must be set to enable the conversion of ADC V_{IN}[13] (V_{REFINT}).

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

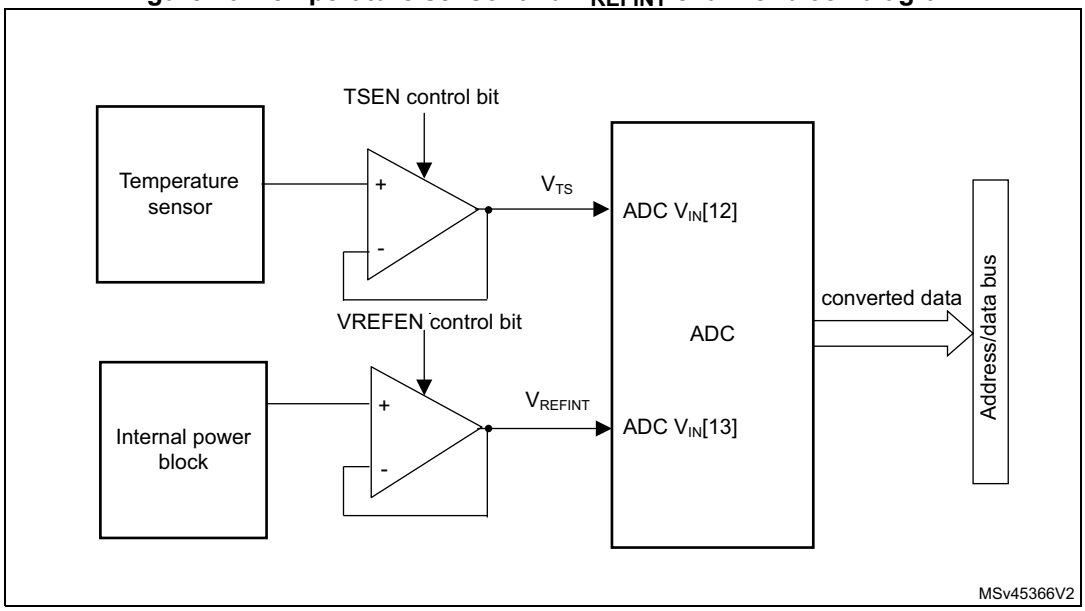
The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. Refer to the datasheet for additional information.

Main features

- Linearity: ±2 °C max., precision depending on calibration

Figure 70. Temperature sensor and V_{REFINT} channel block diagram



Reading the temperature

1. Select the ADC V_{IN}[12] input channel.
2. Select an appropriate sampling time specified in the device datasheet (T_{S_temp}).
3. Set the TSEN bit in the ADC_CCR register to wake up the temperature sensor from power down mode and wait for its stabilization time (t_{START}).
4. Start the ADC conversion by setting the ADSTART bit in the ADC_CR register (or by external trigger).
5. Read the resulting V_{TS} data in the ADC_DR register.
6. Calculate the temperature using the following formula

$$\text{Temperature (in °C)} = \frac{\text{TS\_CAL2\_TEMP} - \text{TS\_CAL1\_TEMP}}{\text{TS\_CAL2} - \text{TS\_CAL1}} \times (\text{TS\_DATA} - \text{TS\_CAL1}) + \text{TS\_CAL1\_TEMP}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP (refer to the datasheet for TS_CAL2 value)
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP (refer to the datasheet for TS_CAL1 value)
- TS_DATA is the actual temperature sensor output value converted by ADC

  Refer to the specific device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Note: The sensor has a startup time after waking from power down mode before it can output V_{TS} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and TSEN bits should be set at the same time.

Calculating the actual V_{REF+} voltage using the internal reference voltage

V_{REF+} voltage may be subject to variation or not precisely known. The embedded internal reference voltage (V_{REFINT}) and its calibration data acquired by the ADC during the manufacturing process at V_{REF+_charac} can be used to evaluate the actual V_{REF+} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+\_Charac} \times VREFINT\_CAL / VREFINT\_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value
- VREFINT_DATA is the actual VREFINT output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the analog power supply and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent of V_{REF+}. For applications where V_{REF+} is known and ADC converted values are right-aligned you can use the following formula to get this absolute value:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL\_SCALE} \times ADC\_DATA_x$$

For applications where V_{REF+} value is not known, you must use the internal voltage reference and V_{REF+} can be replaced by the expression provided in *Section : Calculating the actual V_{REF+} voltage using the internal reference voltage*, resulting in the following formula:

$$V_{CHANNELx} = \frac{V_{REF+\_Charac} \times VREFINT\_CAL \times ADC\_DATA_x}{VREFINT\_DATA \times FULL\_SCALE}$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value
- ADC_DATA_x is the value measured by the ADC on channelx (right-aligned)
- VREFINT_DATA is the actual VREFINT output value converted by the ADC
- full_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, it is 2^{12} - 1 = 4095 or with 8-bit resolution, 2^8 - 1 = 255.

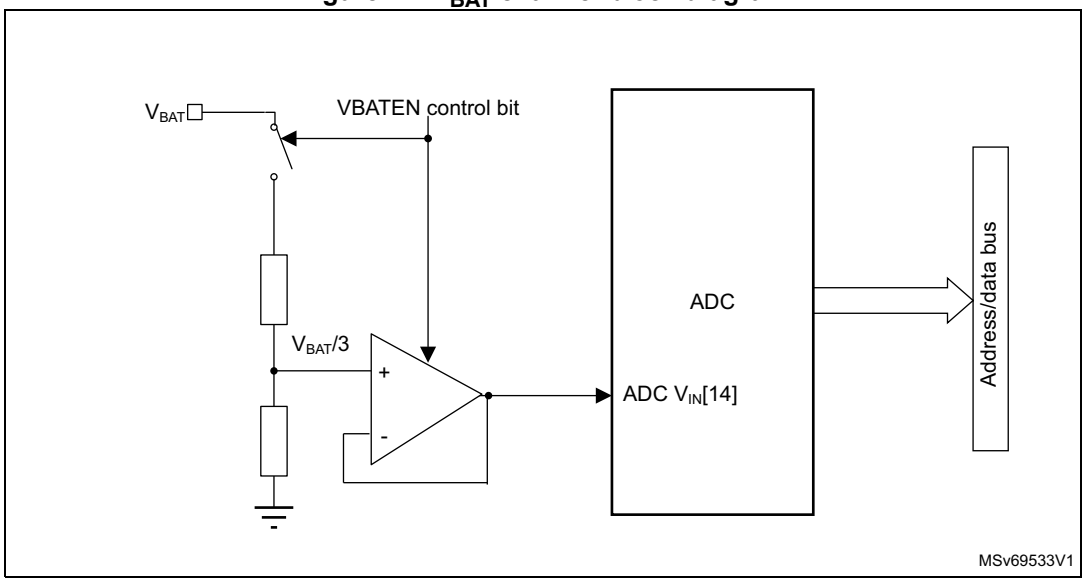
Note: If ADC measurements are done using an output format other than 12 bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

16.10 Battery voltage monitoring

The VBATEN bit in the ADC_CCR register allows the application to measure the backup battery voltage on the VBAT pin. As the V_{BAT} voltage could be higher than V_{REF+}, to ensure

the correct operation of the ADC, the VBAT pin is internally connected to a bridge divider. This bridge is automatically enabled when VBATEN is set, to connect V_{BAT} to the ADC V_{IN}[14] input channel. As a consequence, the converted digital value is V_{BAT}/3. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed for ADC conversion.

Figure 71. V_{BAT} channel block diagram



16.11 ADC interrupts

An interrupt can be generated by any of the following events:

- End Of Calibration (EOCAL flag)
- ADC power-up, when the ADC is ready (ADRDY flag)
- End of any conversion (EOC flag)
- End of a sequence of conversions (EOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2, AWD3 flags)
- When the Channel configuration is ready (CCRDY flag)
- When the end of sampling phase occurs (EOSMP flag)
- when a data overrun occurs (OVR flag)

Separate interrupt enable bits are available for flexibility.

Table 97. ADC interrupts

Interrupt event	Event flag	Enable control bit
End Of Calibration	EOCAL	EOCALIE
ADC ready	ADRDY	ADRDYIE
End of conversion	EOC	EOCIE
End of sequence of conversions	EOS	EOSIE
Analog watchdog 1 status bit is set	AWD1	AWD1IE

Table 97. ADC interrupts (continued)

Interrupt event	Event flag	Enable control bit
Analog watchdog 2 status bit is set	AWD2	AWD2IE
Analog watchdog 3 status bit is set	AWD3	AWD3IE
Channel Configuration Ready	CCRDY	CCRDYIE
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE

16.12 ADC registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

16.12.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CCRDY	Res.	EOCAL	Res.	AWD3	AWD2	AWD1	Res.	Res.	OVR	EOS	EOC	EOSMP	ADRDY
		rc_w1		rc_w1		rc_w1	rc_w1	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **CCRDY**: Channel Configuration Ready flag

This flag bit is set by hardware when the channel configuration is applied after programming to ADC_CHSELR register or changing CHSELRMOD or SCANDIR. It is cleared by software by programming it to it.

0: Channel configuration update not applied.

1: Channel configuration update is applied.

Note: When the software configures the channels (by programming ADC_CHSELR or changing CHSELRMOD or SCANDIR), it must wait until the CCRDY flag rises before configuring again or starting conversions, otherwise the new configuration (or the START bit) is ignored. Once the flag is asserted, if the software needs to configure again the channels, it must clear the CCRDY flag before proceeding with a new configuration.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **EOCAL**: End Of Calibration flag

This bit is set by hardware when calibration is complete. It is cleared by software writing 1 to it.

0: Calibration is not complete

1: Calibration is complete

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD3TR and ADC_AWD3TR registers. It is cleared by software by programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD2TR and ADC_AWD2TR registers. It is cleared by software programming it it.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC_TR1 and ADC_HR1 registers. It is cleared by software by programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog event occurred

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs, meaning that a new conversion has complete while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of sequence flag

This bit is set by hardware at the end of the conversion of a sequence of channels selected by the CHSEL bits. It is cleared by software writing 1 to it.

0: Conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Conversion sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each conversion of a channel when a new data result is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.

0: Channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion, at the end of the sampling phase.It is cleared by software by programming it to ‘1’.

0: Not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

16.12.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CCRDYIE	Res.	EOCALIE	Res.	AWD3IE	AWD2IE	AWD1IE	Res.	Res.	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
		rw		rw		rw	rw	rw			rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **CCRDYIE**: Channel Configuration Ready Interrupt enable

This bit is set and cleared by software to enable/disable the channel configuration ready interrupt.

0: Channel configuration ready interrupt disabled

1: Channel configuration ready interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.

Bit 11 **EOCALIE**: End of calibration interrupt enable

This bit is set and cleared by software to enable/disable the end of calibration interrupt.

0: End of calibration interrupt disabled

1: End of calibration interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the overrun interrupt.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 3 **EOSIE**: End of conversion sequence interrupt enable

This bit is set and cleared by software to enable/disable the end of sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 2 **EOCIE**: End of conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of conversion interrupt.

0: EOC interrupt disabled

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled.

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

16.12.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
											rs		rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC.
It is cleared by hardware after calibration is complete.
0: Calibration complete
1: Write 1 to calibrate the ADC. Read at 1 means that a calibration is in progress.

Note: The software is allowed to set ADCAL only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0, AUTOFF = 0, and ADEN = 0).

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 (ADC enabled and no conversion is ongoing).

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **ADVREGEN**: ADC Voltage Regulator Enable

This bit is set by software, to enable the ADC internal voltage regulator. The voltage regulator output is available after $t_{ADCVREG\_STUP}$.
It is cleared by software to disable the voltage regulator. It can be cleared only if ADEN is et to 0.
0: ADC voltage regulator disabled
1: ADC voltage regulator enabled

Note: The software is allowed to program this bit field only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:5 Reserved, must be kept at reset value.

Bit 4 **ADSTP**: ADC stop conversion command

This bit is set by software to stop and discard an ongoing conversion (ADSTP Command).
It is cleared by hardware when the conversion is effectively discarded and the ADC is ready to accept a new start conversion command.
0: No ADC stop conversion command ongoing
1: Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress.

Note: Setting ADSTP to '1' is only effective when ADSTART = 1 and ADDIS = 0 (ADC is enabled and may be converting and there is no pending request to disable the ADC)

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ADSTART**: ADC start conversion command

This bit is set by software to start ADC conversion. Depending on the EXTEN [1:0] configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– In single conversion mode (CONT = 0, DISCEN = 0), when software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion Sequence (EOS) flag.

– In discontinuous conversion mode(CONT = 0, DISCEN = 1), when the software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion (EOC) flag.

– In all other cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware.

0: No ADC conversion is ongoing.

1: Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

After writing to ADC_CHSELR register or changing CHSELRMOD or SCANDIRW, it is mandatory to wait until CCRDY flag is asserted before setting ADSTART, otherwise, the value written to ADSTART is ignored.

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: No ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: Setting ADDIS to ‘1’ is only effective when ADEN = 1 and ADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable command

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL = 0, ADSTP = 0, ADSTART = 0, ADDIS = 0 and ADEN = 0)

16.12.4 ADC configuration register 1 (ADC_CFGR1)

Address offset: 0x0C

Reset value: 0x0000 0000

The software is allowed to program ADC_CFGR1 only when ADEN is cleared in ADC_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWD1CH[4:0]					Res.	Res.	AWD1EN	AWD1SGL	CHSELRMOD	Res.	Res.	Res.	Res.	DISCEN
	rw	rw	rw	rw	rw			rw	rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTOFF	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	RES[1:0]		SCANDIR	DMACFG	DMAEN
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input Channel 0 monitored by AWD

00001: ADC analog input Channel 1 monitored by AWD

.....

10001: ADC analog input Channel 17 monitored by AWD

Others: Reserved

Note: The channel selected by the AWDCH[4:0] bits must be also set into the CHSELR register.

The software is allowed to write this bit only when ADEN bit is cleared.

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **AWD1EN**: Analog watchdog enable

This bit is set and cleared by software.

0: Analog watchdog **1** disabled

1: Analog watchdog **1** enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bit 22 **AWD1SGL**: Enable the watchdog on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWDCH[4:0] bits or on all the channels

0: Analog watchdog **1** enabled on all channels

1: Analog watchdog **1** enabled on a single channel

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bit 21 **CHSELRMOD**: Mode selection of the ADC_CHSELR register

This bit is set and cleared by software to control the ADC_CHSELR feature:

0: Each bit of the ADC_CHSELR register enables an input

1: ADC_CHSELR register is able to sequence up to 8 channels

Note: The software is allowed to write this bit only when ADEN bit is cleared.

If CCRDY is not yet asserted after channel configuration (writing ADC_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit is ignored.

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **DISCEN**: Discontinuous mode

This bit is set and cleared by software to enable/disable discontinuous mode.

0: Discontinuous mode disabled

1: Discontinuous mode enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADEN bit is cleared.

Bit 15 **AUTOFF**: Auto-off mode

This bit is set and cleared by software to enable/disable auto-off mode.

0: Auto-off mode disabled

1: Auto-off mode enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bit 14 **WAIT**: Wait conversion mode

This bit is set and cleared by software to enable/disable wait conversion mode.

0: Wait conversion mode off

1: Wait conversion mode on

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bit 13 **CONT**: Single / continuous conversion mode

This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADEN bit is cleared.

Bit 12 **OVRMOD**: Overrun management mode

This bit is set and cleared by software and configure the way data overruns are managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection

These bits are set and cleared by software to select the external trigger polarity and enable the trigger.

00: Hardware trigger detection disabled (conversions can be started by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bit 9 Reserved, must be kept at reset value.

Bits 8:6 **EXTSEL[2:0]**: External trigger selection

These bits select the external event used to trigger the start of conversion (refer to *Table 90: External triggers* for details):

000: TRG0
001: TRG1
010: TRG2
011: TRG3
100: TRG4
101: TRG5
110: TRG6
111: TRG7

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to *Figure 57: Data alignment and resolution (oversampling disabled: OVSE = 0) on page 447*

0: Right alignment
1: Left alignment

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12 bits
01: 10 bits
10: 8 bits
11: 6 bits

Note: The software is allowed to write these bits only when ADEN is cleared.

Bit 2 **SCANDIR**: Scan sequence direction

This bit is set and cleared by software to select the direction in which the channels is scanned in the sequence. It is effective only if CHSELMOD bit is cleared.

0: Upward scan (from CHSEL0 to CHSEL17)
1: Backward scan (from CHSEL17 to CHSEL0)

Note: The software is allowed to write this bit only when ADEN bit is cleared.

If CCRDY is not yet asserted after channel configuration (writing ADC_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit is ignored.

Bit 1 **DMACFG**: Direct memory access configuration

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

0: DMA one shot mode selected
1: DMA circular mode selected

For more details, refer to *Section 16.5.5: Managing converted data using the DMA on page 449*

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows the DMA controller to be used to manage automatically the converted data. For more details, refer to *Section 16.5.5: Managing converted data using the DMA on page 449*.

0: DMA disabled
1: DMA enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared.

16.12.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

The software is allowed to program ADC_CFGR2 only when ADEN is cleared in ADC_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKMODE[1:0]		LFTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TOVS	OVSS[3:0]				OVSR[2:0]			Res.	OVSE
						rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:30 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define how the analog ADC is clocked:

00: ADCCLK (Asynchronous clock mode), generated at product level (refer to RCC section)

01: PCLK/2 (Synchronous clock mode)

10: PCLK/4 (Synchronous clock mode)

11: PCLK (Synchronous clock mode). This configuration must be enabled only if PCLK has a 50% duty clock cycle (APB prescaler configured inside the RCC must be bypassed and the system clock must by 50% duty cycle)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 **LFTRIG**: Low frequency trigger mode enable

This bit is set and cleared by software.

0: Low Frequency Trigger Mode disabled

1: Low Frequency Trigger Mode enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared.T

Bits 28:10 Reserved, must be kept at reset value.

Bit 9 **TOVS**: Triggered Oversampling

This bit is set and cleared by software.

0: All oversampled conversions for a channel are done consecutively after a trigger

1: Each oversampled conversion for a channel needs a trigger

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bits 8:5 **OVSS[3:0]**: Oversampling shift
This bit is set and cleared by software.
0000: No shift
0001: Shift 1-bit
0010: Shift 2-bits
0011: Shift 3-bits
0100: Shift 4-bits
0101: Shift 5-bits
0110: Shift 6-bits
0111: Shift 7-bits
1000: Shift 8-bits
Others: Reserved
Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bits 4:2 **OVSR[2:0]**: Oversampling ratio
This bit filed defines the number of oversampling ratio.
000: 2x
001: 4x
010: 8x
011: 16x
100: 32x
101: 64x
110: 128x
111: 256x
Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **OVSE**: Oversampler Enable
This bit is set and cleared by software.
0: Oversampler disabled
1: Oversampler enabled
Note: The software is allowed to write this bit only when ADEN bit is cleared.

16.12.6 ADC sampling time register (ADC_SMPR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMPSEL17	SMPSEL16	SMPSEL15	SMPSEL14	SMPSEL13	SMPSEL12	SMPSEL11	SMPSEL10	SMPSEL9	SMPSEL8
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPSEL7	SMPSEL6	SMPSEL5	SMPSEL4	SMPSEL3	SMPSEL2	SMPSEL1	SMPSEL0	Res.	SMP2[2:0]			Res.	SMP1[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:8 **SMPSELx*:*** Channel-x sampling time selection (x = 17 to 0)

These bits are written by software to define which sampling time is used.

0: Sampling time of CHANNELx use the setting of SMP1[2:0] register.

1: Sampling time of CHANNELx use the setting of SMP2[2:0] register.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SMP2[2:0]:** Sampling time selection 2

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 160.5 ADC clock cycles

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMP1[2:0]:** *Sampling time selection* 1

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 160.5 ADC clock cycles

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

16.12.7 ADC watchdog threshold register (ADC_AWD1TR)

Address offset: 0x20

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog.
Refer to *Section 16.7: Analog window watchdogs on page 452*.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT1[11:0]**: Analog watchdog *1* lower threshold
These bits are written by software to define the lower threshold for the analog watchdog.
Refer to *Section 16.7: Analog window watchdogs on page 452*.

16.12.8 ADC watchdog threshold register (ADC_AWD2TR)

Address offset: 0x24

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT2[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT2[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT2[11:0]**: Analog watchdog 2 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog.
Refer to *Section 16.7: Analog window watchdogs on page 452*.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT2[11:0]**: Analog watchdog 2 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog.
Refer to *Section 16.7: Analog window watchdogs on page 452*.

16.12.9 ADC channel selection register (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

– Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFGR1). Refer to the current section.

– ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFGR1). Refer to next section.

CHSELRMOD = 0 in ADC_CFGR1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHSEL 17	CHSEL 16
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHSEL 15	CHSEL 14	CHSEL 13	CHSEL 12	CHSEL 11	CHSEL 10	CHSEL 9	CHSEL 8	CHSEL 7	CHSEL 6	CHSEL 5	CHSEL 4	CHSEL 3	CHSEL 2	CHSEL 1	CHSEL 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **CHSEL[17:0]**: Channel-x selection

These bits are written by software and define which channels are part of the sequence of channels to be converted. Refer to *Figure 49: ADC connectivity* for ADC inputs connected to external channels and internal sources.

0: Input Channel-x is not selected for conversion

1: Input Channel-x is selected for conversion

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

If CCRDY is not yet asserted after channel configuration (writing ADC_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit is ignored.

16.12.10 ADC channel selection register [alternate] (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

– Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFGR1). Refer to the current previous section.

– ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFGR1). Refer to this section.

CHSELRMOD = 1 in ADC_CFGR1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SQ8[3:0]				SQ7[3:0]				SQ6[3:0]				SQ5[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[3:0]				SQ3[3:0]				SQ2[3:0]				SQ1[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **SQ8[3:0]**: 8th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates the end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

0000: CH0

0001: CH1

...

1100: CH12

1101: CH13

1110: CH14

1111: No channel selected (End of sequence)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:24 **SQ7[3:0]**: 7th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:20 **SQ6[3:0]**: 6th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 19:16 **SQ5[3:0]**: 5th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:12 **SQ4[3:0]**: 4th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 11:8 **SQ3[3:0]**: 3rd conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 7:4 **SQ2[3:0]**: 2nd conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 3:0 **SQ1[3:0]**: 1st conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

16.12.11 ADC watchdog threshold register (ADC_AWD3TR)

Address offset: 0x2C

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT3[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT3[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT3[11:0]**: *Analog watchdog 3 higher threshold*

These bits are written by software to define the higher threshold for the analog watchdog.
Refer to *Section 16.7: Analog window watchdogs on page 452*.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT3[11:0]**: Analog watchdog *3*lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.
Refer to *Section 16.7: Analog window watchdogs on page 452*.

16.12.12 ADC data register (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DATA[15:0]**: Converted data

These bits are read-only. They contain the conversion result from the last converted channel. The data are left- or right-aligned as shown in *Figure 57: Data alignment and resolution (oversampling disabled: OVSE = 0) on page 447*.

Just after a calibration is complete, DATA[6:0] contains the calibration factor.

16.12.13 ADC Analog Watchdog 2 Configuration register (ADC_AWD2CR)

Address offset: 0xA0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2 CH17	AWD2 CH16
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2 CH15	AWD2 CH14	AWD2 CH13	AWD2 CH12	AWD2 CH11	AWD2 CH10	AWD2 CH9	AWD2 CH8	AWD2 CH7	AWD2 CH6	AWD2 CH5	AWD2 CH4	AWD2 CH3	AWD2 CH2	AWD2 CH1	AWD2 CH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **AWD2CH[17:0]:** Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 2 (AWD2).

0: ADC analog channel-x is not monitored by AWD2

1: ADC analog channel-x is monitored by AWD2

Note: The channels selected through ADC_AWD2CR must be also configured into the ADC_CHSELR registers. The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

16.12.14 ADC Analog Watchdog 3 Configuration register (ADC_AWD3CR)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3 CH17	AWD3 CH16
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3 CH15	AWD3 CH14	AWD3 CH13	AWD3 CH12	AWD3 CH11	AWD3 CH10	AWD3 CH9	AWD3 CH8	AWD3 CH7	AWD3 CH6	AWD3 CH5	AWD3 CH4	AWD3 CH3	AWD3 CH2	AWD3 CH1	AWD3 CH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **AWD3CH[17:0]:** Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 3 (AWD3).

0: ADC analog channel-x is not monitored by AWD3

1: ADC analog channel-x is monitored by AWD3

Note: The channels selected through ADC_AWD3CR must be also configured into the ADC_CHSELR registers. The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

16.12.15 ADC Calibration factor (ADC_CALFACT)

Address offset: 0xB4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT[6:0]**: Calibration factor

These bits are written by hardware or by software.

– Once a calibration is complete, they are updated by hardware with the calibration factors.

– Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new conversion is launched.

– Just after a calibration is complete, DATA[6:0] contains the calibration factor.

Note: Software can write these bits only when ADEN=1 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

16.12.16 ADC common configuration register (ADC_CCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBAT EN	TSEN	VREF EN	PRESC[3:0]				Res.	Res.
							rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: V_{BAT} enable

This bit is set and cleared by software to enable/disable the V_{BAT} channel.

0: V_{BAT} channel disabled
1: V_{BAT} channel enabled

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing)

Bit 23 **TSEN**: Temperature sensor enable

This bit is set and cleared by software to enable/disable the temperature sensor.

0: Temperature sensor disabled
1: Temperature sensor enabled

Note: Software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT}.

0: V_{REFINT} disabled
1: V_{REFINT} enabled

Note: Software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 21:18 **PRESC[3:0]:** ADC prescaler

Set and cleared by software to select the frequency of the clock to the ADC.
0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
Other: Reserved

Note: Software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 17:0 Reserved, must be kept at reset value.

16.13 ADC register map

The following table summarizes the ADC registers.

Table 98. ADC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRDY.	Res.	EOCAL	Res.	AWD3	AWD2	AWD1	Res.	Res.	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																			0		0		0	0	0			0	0	0	0	0

Table 98. ADC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04	**ADC_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRDYIE.	Res.	EOCALIE	Res.	AWD3IE	AWD2IE	AWD1IE	Res.	Res.	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																			0		0		0	0	0			0	0	0	0	0
0x08	**ADC_CR**	ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
	Reset value	0			0																								0		0	0	0
0x0C	**ADC_CFGR1**	Res.	AWDCH[4:0]					Res.	Res.	AWD1EN	AWD1SGL	CHSELRMOD	Res.	Res.	Res.	Res.	DISCEN	AUTOFF	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL [2:0]			ALIGN	RES [1:0]		SCANDIR	DMACFG	DMAEN
	Reset value		0	0	0	0	0			0	0	0					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x10	**ADC_CFGR2**	CKMODE[1:0]		LFTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOVS	OVSS[3:0]				OVSR[2:0]			Res.	OVSE
	Reset value	0	0	0																			0	0	0	0	0	0	0	0	0	0	0
0x14	**ADC_SMPR**	Res.	Res.	Res.	Res.	Res.	Res.	SMPSEL17	SMPSEL16	SMPSEL15	SMPSEL14	SMPSEL13	SMPSEL12	SMPSEL11	SMPSEL10	SMPSEL9	SMPSEL8	SMPSEL7	SMPSEL6	SMPSEL5	SMPSEL4	SMPSEL3	SMPSEL2	SMPSEL1	SMPSEL0	Res.	SMP2 [2:0]		Res.	SMP1 [2:0]			
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0		0	0	0
0x18	Reserved	Reserved																															
0x1C	Reserved	Reserved																															
0x20	**ADC_AWD1TR**	Res.	Res.	Res.	Res.	HT1[11:0]												Res.	Res.	Res.	Res.	LT1[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADC_AWD2TR**	Res.	Res.	Res.	Res.	HT2[11:0]												Res.	Res.	Res.	Res.	LT2[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x28	**ADC_CHSELR** (CHSELRMOD= 0)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res	Res	CHSEL17	CHSEL16	CHSEL15	CHSEL14	CHSEL13	CHSEL12	CHSEL11	CHSEL10	CHSEL9	CHSEL8	CHSEL7	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**ADC_CHSELR** (CHSELRMOD= 1)	SQ8[3:0]				SQ7[3:0]				SQ6[3:0]				SQ5[3:0]				SQ4[3:0]				SQ3[3:0]				SQ2[3:0]				SQ1[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**ADC_AWD3TR**	Res.	Res.	Res.	Res.	HT3[11:0]												Res.	Res.	Res.	Res.	LT3[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x30 0x34 0x38 0x3C	Reserved	Reserved																															
0x40	**ADC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	Reserved	Reserved																															
0xA0	**ADC_AWD2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res	Res	Res	AWD2CH17	AWD2CH16	AWD2CH15	AWD2CH14	AWD2CH13	AWD2CH12	AWD2CH11	AWD2CH10	AWD2CH9	AWD2CH8	AWD2CH7	AWD2CH6	AWD2CH5	AWD2CH4	AWD2CH3	AWD2CH2	AWD2CH1	AWD2CH0
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 98. ADC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA4	**ADC_AWD3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH17	AWD3CH16	AWD3CH15	AWD3CH14	AWD3CH13	AWD3CH12	AWD3CH11	AWD3CH10	AWD3CH9	AWD3CH8	AWD3CH7	AWD3CH6	AWD3CH5	AWD3CH4	AWD3CH3	AWD3CH2	AWD3CH1	AWD3CH0
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	Reserved	Reserved																															
0xB4	**ADC_CALFACT**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CALFACT[6:0]						
	Reset value																										0	0	0	0	0	0	0
...	Reserved	Reserved																															
0x308	**ADC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	PRESC3	PRESC2	PRESC1	PRESC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0	0	0	0	0	0																		

Refer to *Section 2.4 on page 62* for the register boundary addresses.

17 Digital-to-analog converter (DAC)

17.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC features one single channel. An input reference pin, V_{REF+} (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to *voltage reference buffer (VREFBUF)* section.

The DACx_OUT1 pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the Sample and hold mode.

17.2 DAC main features

The DAC main features are the following (see *Figure 72: DAC block diagram*)

- One DAC interface
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Single DAC channel
- DMA capability for each channel including DMA underrun error detection
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- The DAC output can be disconnected from the DACx_OUT1 output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from V_{REF+} pin or internal VREFBUF reference

Figure 72 shows the block diagram of a DAC channel and *Table 100* gives the pin description.

17.3 DAC implementation

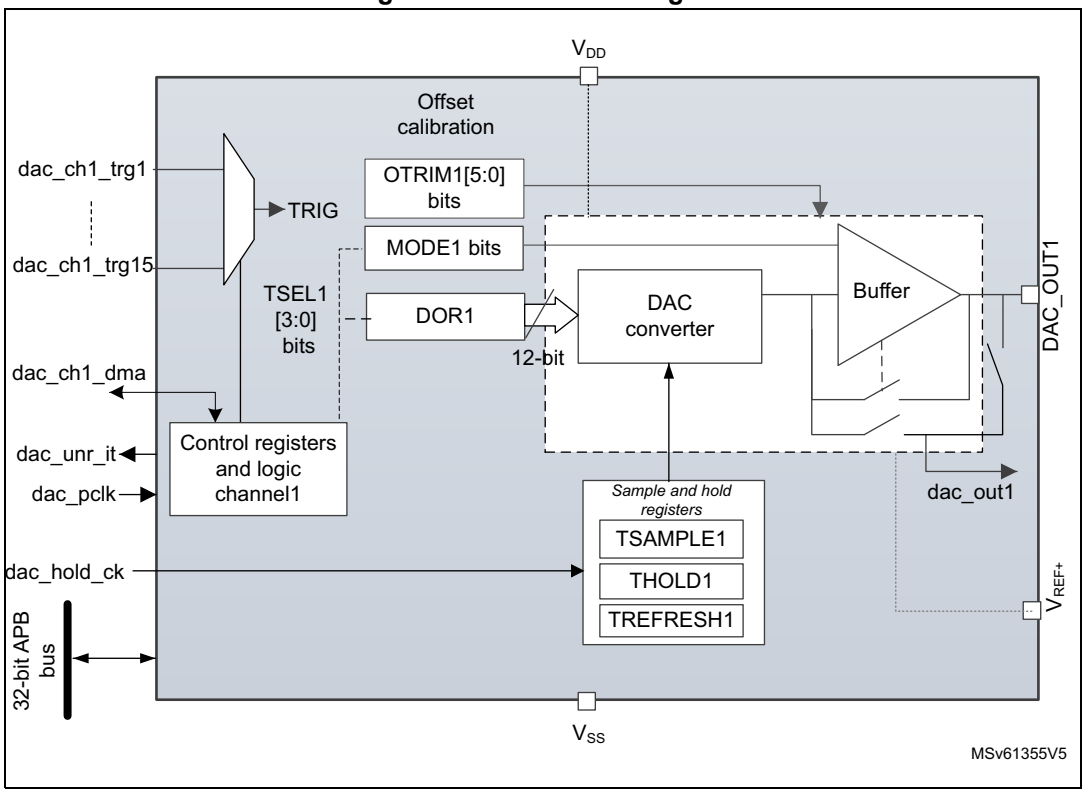
Table 99. DAC features

DAC features	DAC
Dual channel	-
Output buffer	X
I/O connection	DAC_OUT1 to PA10
Maximum sampling time	1 Msps
Autonomous mode	-
VREF+ pin	X

17.4 DAC functional description

17.4.1 DAC block diagram

Figure 72. DAC block diagram



1. MODE1 bits in the DAC_MCR control the output mode and the switching between the Normal mode in buffer/unbuffered configuration and the Sample and hold mode.

17.4.2 DAC pins and internal signals

The DAC includes:

- One output channel
- The DACx_OUT1 can be disconnected from the output pin and used as an ordinary GPIO
- The dac_out1 can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUT1 output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The Sample and hold block and its associated registers can run in Stop mode using the LSI clock source (dac_hold_ck).

Table 100. DAC input/output pins

Pin name	Signal type	Remarks
VREF+	Input, analog reference positive	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet)
VDD	Input, analog supply	Analog power supply
VSS	Input, analog supply ground	Ground for analog power supply
DACx_OUT1	Analog output signal	DACx channel1 analog output

Table 101. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs/acknowledge
dac_unr_it	Output	DAC underrun interrupt
dac_pclk	Input	DAC peripheral clock
dac_hold_ck	Input	DAC low-power clock used in Sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals

Table 102. DAC interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi	LSI clock selected in the RCC
dac_ch1_trg1	tim1_trgo	Internal signal from on-chip timers TIM1_TGO_CKTIM

Table 102. DAC interconnection (continued)

Signal name	Source	Source type
dac_ch1_trg2	tim2_trgo	Internal signal from on-chip timers TIM2_TGO_CKTIM
dac_ch1_trg11	lptim1_out	Internal signal from on-chip timers LPTIM1_OUT
dac_ch1_trg12	lptim2_out	Internal signal from on-chip timers LPTIM2_OUT
dac_ch1_trg13	lptim3_out	Internal signal from on-chip timers LPTIM3_OUT
dac_ch1_trg14	exti9	External pin EXTI[9]

17.4.3 DAC channel enable

The DAC channel can be powered on by setting its corresponding EN1 bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

Note: The EN1 bit enables the analog DAC channel1 only. The DAC channel1 digital interface is enabled even if the EN1 bit is reset.

17.4.4 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

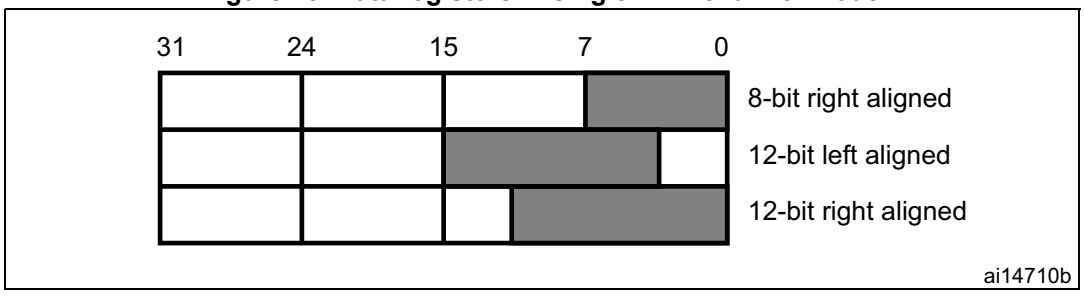
  – 8-bit right alignment: the software has to load data into the DAC_DHR8R1[7:0] bits (stored into the DHR1[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12L1 [15:4] bits (stored into the DHR1[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12R1 [11:0] bits (stored into the DHR1[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHR1 (data holding registerx, which are internal non-memory-mapped registers). The DHR1 register is then loaded into the DOR1 register either automatically, by software trigger or by an external event trigger.

Figure 73. Data registers in single DAC channel mode



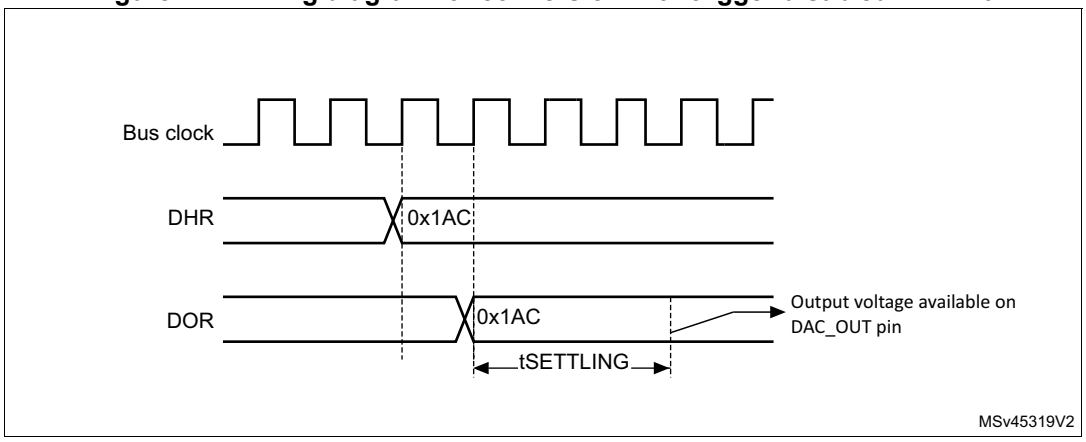
17.4.5 DAC conversion

The DAC_DOR1 cannot be written directly and any data transfer to the DAC channel1 must be performed by loading the DAC_DHR1 register (write operation to DAC_DHR8R1, DAC_DHR12L1, DAC_DHR12R1, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHR1 register are automatically transferred to the DAC_DOR1 register after one dac_pclk clock cycle, if no hardware trigger is selected (TEN1 bit in DAC_CR register is reset). However, when a hardware trigger is selected (TEN1 bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_pclk clock cycles after the trigger signal.

When DAC_DOR1 is loaded with the DAC_DHR1 contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

Figure 74. Timing diagram for conversion with trigger disabled TEN = 0



17.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltage on the DAC channel pin is determined by the following equation:

$$\text{DACoutput} = V_{REF} \times \frac{DOR}{4096}$$

17.4.7 DAC trigger selection

If the TEN1 control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSEL1[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSEL1[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in *Section 17.4.2: DAC pins and internal signals*.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHR1 register are transferred into the DAC_DOR1 register. The DAC_DOR1 register is updated three dac_pclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DOR1 register has been loaded with the DAC_DHR1 register contents.

Note: TSEL1[3:0] bit cannot be changed when the EN1 bit is set.

When software trigger is selected, the transfer from the DAC_DHR1 register to the DAC_DOR1 register takes only one dac_pclk clock cycle.

17.4.8 DMA requests

The DAC channel has a DMA capability. One DMA channel is used to service DAC channel DMA request.

When an external trigger (but not a software trigger) occurs while the DMAEN1 bit is set, the value of the DAC_DHR1 register is transferred into the DAC_DOR1 register when the transfer is complete, and a DMA request is generated.

As DAC_DHR1 to DAC_DOR1 data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHR1 before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channel1 underrun flag DMAUDR1 in the DAC_SR register is set, reporting the error condition. The DAC channel1 continues to convert old data.

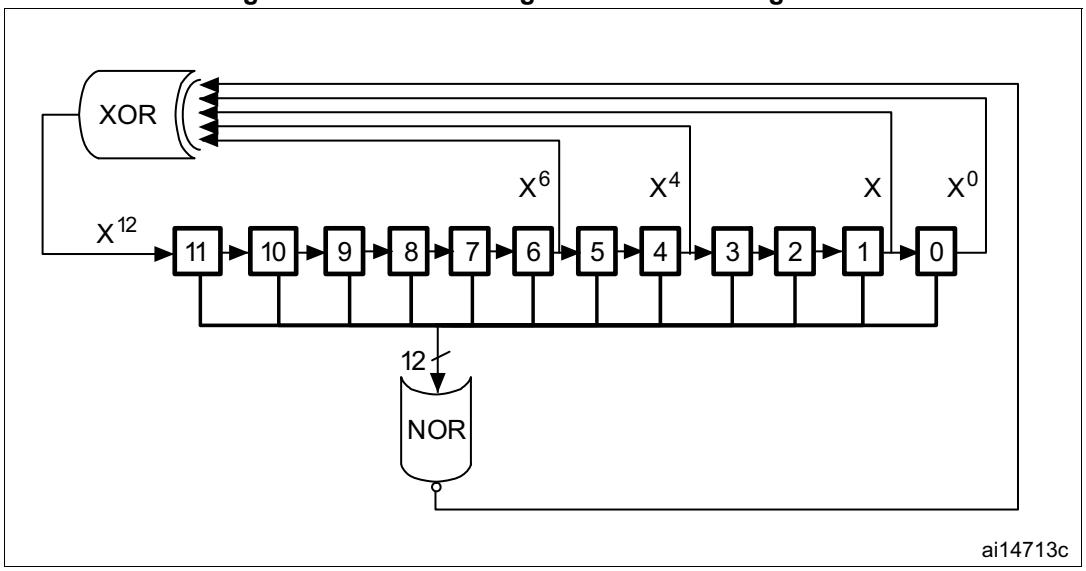
The software must clear the DMAUDR1 flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channel1 to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For DAC channel1, an interrupt is also generated if its corresponding DMAUDRIE1 bit in the DAC_CR register is enabled.

17.4.9 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVE1[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_pclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 75. DAC LFSR register calculation algorithm

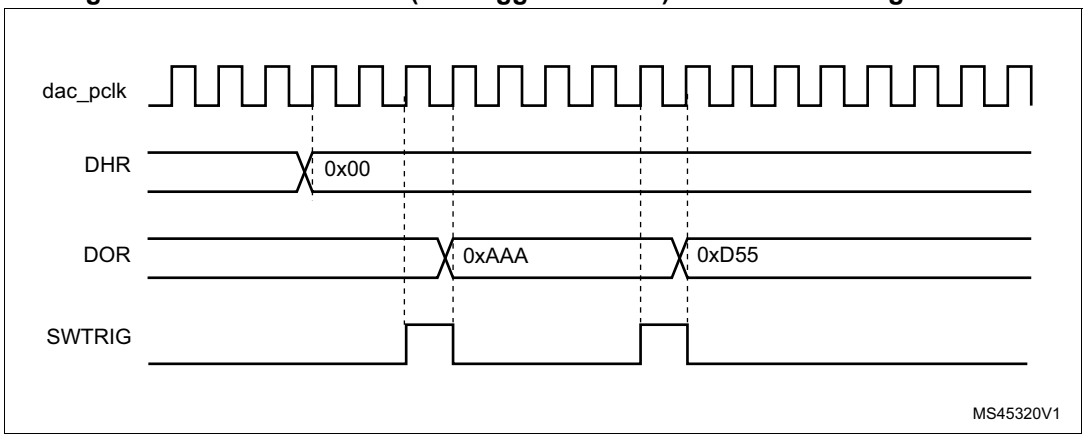


The LFSR value, that may be masked partially or totally by means of the MAMP1[3:0] bits in the DAC_CR register, is added up to the DAC_DHR1 contents without overflow and this value is then transferred into the DAC_DOR1 register.

If LFSR is 0x0000, a ‘1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVE1[1:0] bits.

Figure 76. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TEN1 bit in the DAC_CR register.

17.4.10 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVE1[1:0] to 10”. The amplitude is configured through the MAMP1[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_pclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHR1 register without overflow and the sum is transferred into the DAC_DOR1 register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMP1[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVE1[1:0] bits.

Figure 77. DAC triangle wave generation

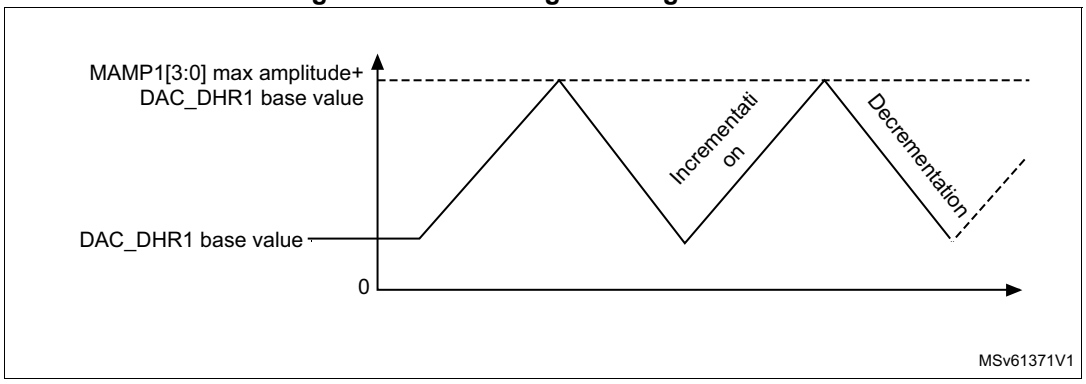
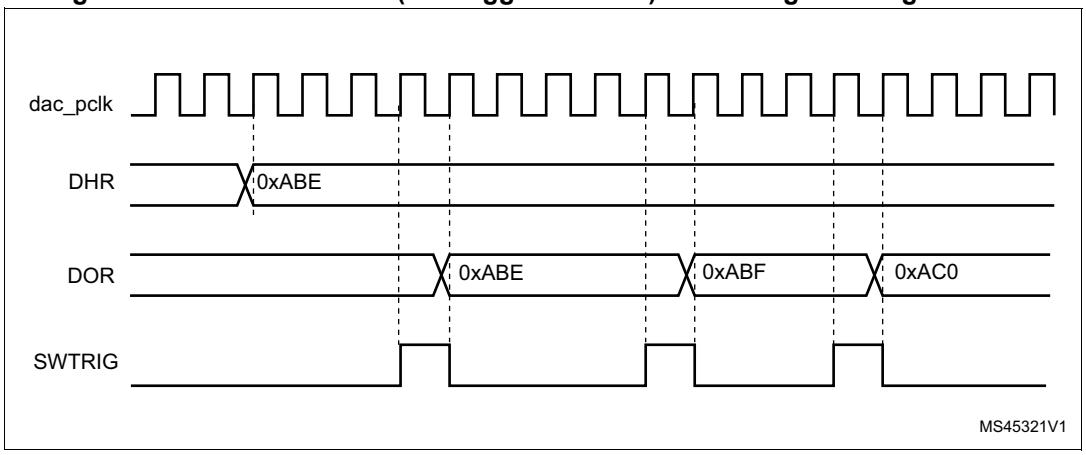


Figure 78. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TEN1 bit in the DAC_CR register.

The MAMP1[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

17.4.11 DAC channel modes

The DAC channel can be configured in Normal mode or Sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In Normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUT1 pin interconnections.

To enable the output buffer, the MODE1[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODE1[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In Sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI low-speed clock (dac_hold_ck) in addition to the dac_pclk clock, allowing the DAC channel to be used in deep low power modes such as Stop mode.

The LSI low-speed clock (dac_hold_ck) must not be stopped when the Sample and hold mode is enabled.

The sample/hold mode operations can be divided into 3 phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLE1[9:0] bits in DAC_SHSR1 register. During the write of the TSAMPLE1[9:0] bits, the BWST1 bit in DAC_SR register is set to 1 to synchronize between both clocks domains (APB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLD1[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESH1[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI ~32 KHz is selected:

12 cycles are required for sample phase: TSAMPLE1[9:0] = 11,

62 cycles are required for hold phase: THOLD1[9:0] = 62,

and 4 cycles are required for refresh period: TREFRESH1[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus Normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 103. Sample and refresh timings

Buffer State	t_{SAMP}(1)(2)	$t_{REFRESH}$(2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in Sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

C_{SH} = 100 nF

V_{DD} = 3.0 V

Sampling phase:

t_{SAMP} = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms

(where R_{BON} = 2 kΩ)

Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs

(where N_{LSB} = 10 (10 LSB drop during the hold phase)

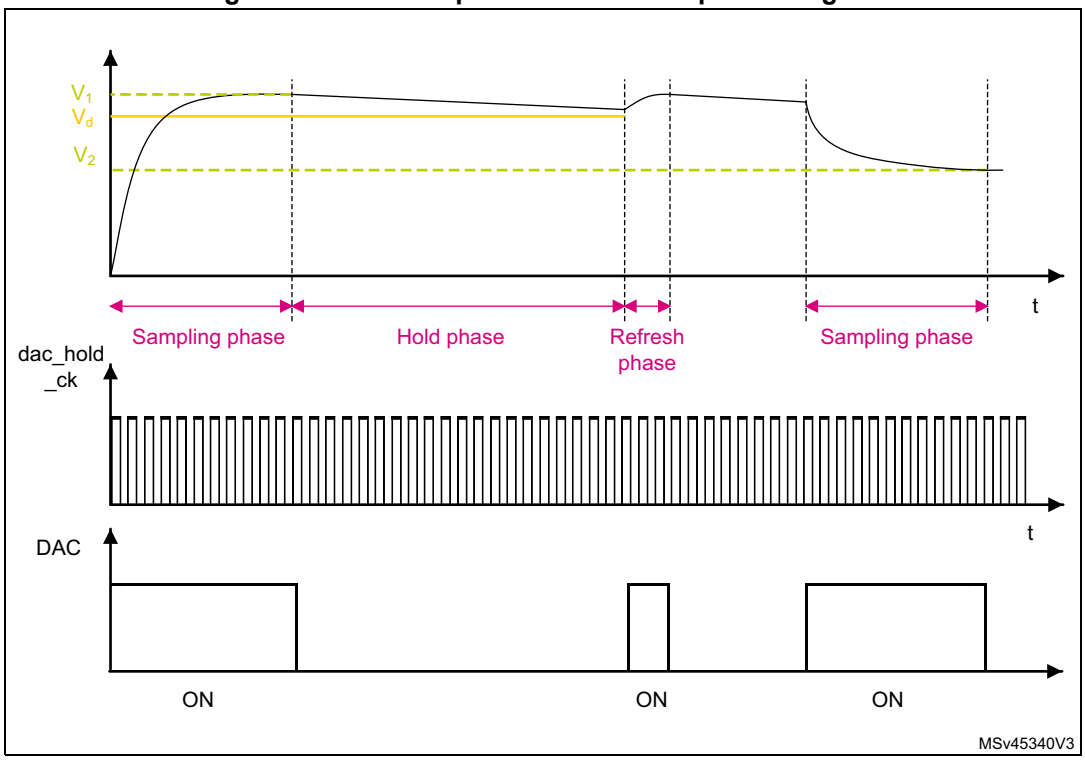
Hold phase:

D_v = i_{leak} * t_{hold} / C_{SH} = 0.0073 V (10 LSB of 12bit at 3 V)

i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)

t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 79. DAC Sample and hold mode phase diagram



Like in Normal mode, the Sample and hold mode has different configurations.

To enable the output buffer, MODE1[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODE1[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

  When MODE1[2:0] bits are equal to 111, an internal capacitor, C_{Lint}, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All Sample and hold phases are interruptible, and any change in DAC_DHR1 immediately triggers a new sample phase.

Table 104. Channel output modes summary

MODE1[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (such as comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (such as comparators)

Table 104. Channel output modes summary (continued)

MODE1[2:0]			Mode	Buffer	Output connections
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

17.4.12 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = ((D / 2^N) \times G \times V_{ref}) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{ref} is the nominal full-scale voltage, and V_{os} is the offset voltage. For an ideal DAC channel, G = 1 and V_{os} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{os}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channel is operating with buffer enabled (MODE1[2:0] = 000b or 001b or 100b or 101b). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to VREF+/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAG1 bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIM1[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, VREF+ values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the Nominal factory trimming conditions

In addition, when V_{DD} is removed (example the device enters in STANDBY or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to EN1 bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODE1[2:0] = 000b or 001b or 100b or 101b.
3. Start the DAC channel calibration, by setting the CEN1 bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIM1[4:0] bits, starting by 00000b.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAG1 bit in DAC_SR is set to 1.
   d) If CAL_FLAG1 is set to 1, the OTRIM1[4:0] trimming code is found and can be used during *device* operation to compensate the output value, else increment OTRIM1[4:0] and repeat sub-steps from (a) to (d) again.

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIM1[4:0] bits in a faster way.

The commutation/toggle of CAL_FLAG1 bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIM1[4:0] bits in DAC_CCR register.

Note: A t_{TRIM} delay must be respected between the write to the OTRIM1[4:0] bits and the read of the CAL_FLAG1 bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If V_{DDA}, VREF+ and temperature conditions do not change during device operation while it enters more often in standby and VBAT mode, the software may store the OTRIM1[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CEN1 bit is set, it is not allowed to set EN1 bit.

17.4.13 DAC channel conversion modes

Four conversion modes are possible.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TEN1.
2. Configure the trigger sources by setting different values in the TSEL1[3:0] bits.
3. Load the DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_pclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TEN1.
2. Configure the trigger sources by setting different values in the TSEL1[3:0] bits.
3. Configure the DAC channel WAVE1[1:0] bits as 01 and the same LFSR mask value in the MAMP1[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bits, TEN1.
2. Configure the trigger sources by setting different values in the TSEL1[3:0] bits.
3. Configure the DAC channel WAVE1[1:0] bits as 1x and the same maximum amplitude value in the MAMP1[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel trigger arrives, the DAC channel triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel triangle counter is then updated.

Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the trigger sources by setting different values in STRSTTRIGSEL1[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the DAC channel WAVE1[1:0] bits to 11 and set the same STRSTDATA1[11:0], STINCDATA1[15:0] and STDIR1 values for each register.

When a DAC channel trigger arrives, the DAC channel sawtooth counter updates the DHR1 register and transfers it into DAC_DOR1 (three APB clock cycles later).

17.5 DAC in low-power modes

Table 105. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA
LPRun	No effect.
LPSleep	No effect. DAC used with DMA.

Table 105. Effect of low-power modes on DAC (continued)

Mode	Description
Stop 0 / Stop 1	The DAC remains active with a static value if the Sample and hold mode is selected using LSI clock.
Stop 2	The DAC registers content is lost and must be reinitialized after exiting Stop 2. The DAC must be disabled before entering Stop 2.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

17.6 DAC interrupts

Table 106. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
DAC	DMA underrun	DMAUDR1	DMAUDRIE1	Write DMAUDRx = 1	Yes	No	No

17.7 DAC registers

Refer to *Section 1 on page 55* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

17.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3]	TSEL1[2]	TSEL1[1]	TSEL1[0]	TEN1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in Normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.
00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled
Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection

These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trg1
0010: dac_ch1_trg2
...
1111: dac_ch1_trg15
Refer to the trigger selection tables in *Section 17.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_pclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_pclk clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_pclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

17.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG1
															w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

17.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

17.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

17.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

17.7.6 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

17.7.7 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

17.7.8 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

17.7.9 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output

These bits are read-only, they contain data output for DAC channel1.

17.7.10 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_ FLAG1	DMAU DR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after Sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag
This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel1
1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bits 10:0 Reserved, must be kept at reset value.

17.7.11 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

17.7.12 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
													rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel1 mode:

– DAC channel1 in Normal mode

000: DAC channel1 is connected to external pin with Buffer enabled

001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled

010: DAC channel1 is connected to external pin with Buffer disabled

011: DAC channel1 is connected to on chip peripherals with Buffer disabled

– DAC channel1 in sample & hold mode

100: DAC channel1 is connected to external pin with Buffer enabled

101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel1 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN1 = 0.

17.7.13 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in Sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1 = 1, the write operation is ignored.

*Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI clock period.*

17.7.14 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in Sample and hold mode)
Hold time = (THOLD[9:0]) x LSI clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

17.7.15 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in Sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

17.7.16 DAC register map

Table 107 summarizes the DAC registers.

Table 107. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**DAC_CR**	Res.	Res.	Res.	Res.	Res.						Res.							CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL13	TSEL12	TSEL11	TSEL10	TEN1	EN1
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**DAC_ SWTRGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG1
	Reset value																																0
0x08	**DAC_ DHR12R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**DAC_ DHR12L1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	**DAC_ DHR8R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x20	**DAC_ DHR12RD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**DAC_ DHR12LD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x28	**DAC_ DHR8RD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x2C	**DAC_ DOR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x34	**DAC_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x38	**DAC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
	Reset value																												X	X	X	X	X
0x3C	**DAC_MCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value																														0	0	0
0x40	**DAC_ SHSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	**DAC_ SHHR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	1

Table 107. DAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4C	DAC_SHRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value																									0	0	0	0	0	0	0	1

Refer to *Section 2.4 on page 62* for the register boundary addresses.

18 Voltage reference buffer (VREFBUF)

18.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for ADC, DAC and also as voltage reference for external components through the VREF+ pin.When the VREF+ pin is double-bonded with VDDA pin in a package, the voltage reference buffer is not available and must be kept disabled (refer to datasheet for packages pinout description).

18.2 VREFBUF functional description

The internal voltage reference buffer supports two voltages[a], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 0: V_{REF_OUT1} around 2.048 V.
- VRS = 1: V_{REF_OUT2} around 2.5 V.

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 108. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – V_{REF+} pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer off – V_{REF+} pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – V_{REF+} pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

18.3 VREFBUF registers

18.3.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRR	VRS	HIZ	ENVR
												r	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **VRR:** Voltage reference buffer ready
0: the voltage reference buffer output is not ready.
1: the voltage reference buffer output reached the requested level.

Bit 2 **VRS**: Voltage reference scale
This bit selects the value generated by the voltage reference buffer.
0: Voltage reference set to V_{REF_OUT1} (around 2.048 V).
1: Voltage reference set to V_{REF_OUT2} (around 2.5 V).

Bit 1 **HIZ:** High impedance mode
This bit controls the analog switch to connect or not the V_{REF+} pin.
0: V_{REF+} pin is internally connected to the voltage reference buffer output.
1: V_{REF+} pin is high impedance.
Refer to *Table 108: VREF buffer modes* for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR:** Voltage reference buffer mode enable
This bit is used to enable the voltage reference buffer mode.
0: Internal voltage reference mode disable (external voltage reference mode).
1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

18.3.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TRIM[5:0]**: Trimming code

These bits are automatically initialized after reset with the trimming value stored in the Flash memory during the production test. Writing into these bits allows the tuning of the internal reference buffer voltage.

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

18.3.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 109. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**VREFBUF_CSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRR	VRS	HIZ	ENVR
	Reset value																													0	0	1	0
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

Refer to *Section 2.4* for the register boundary addresses.

19 Comparator (COMP)

19.1 COMP introduction

The device embeds two ultra-low-power comparators (COMP1 and COMP2).

These comparators can be used for a variety of functions including the following:

- wake up from low-power mode triggered by an analog signal
- analog signal conditioning
- cycle-by-cycle current control loop when combined with a PWM output from a timer

19.2 COMP main features

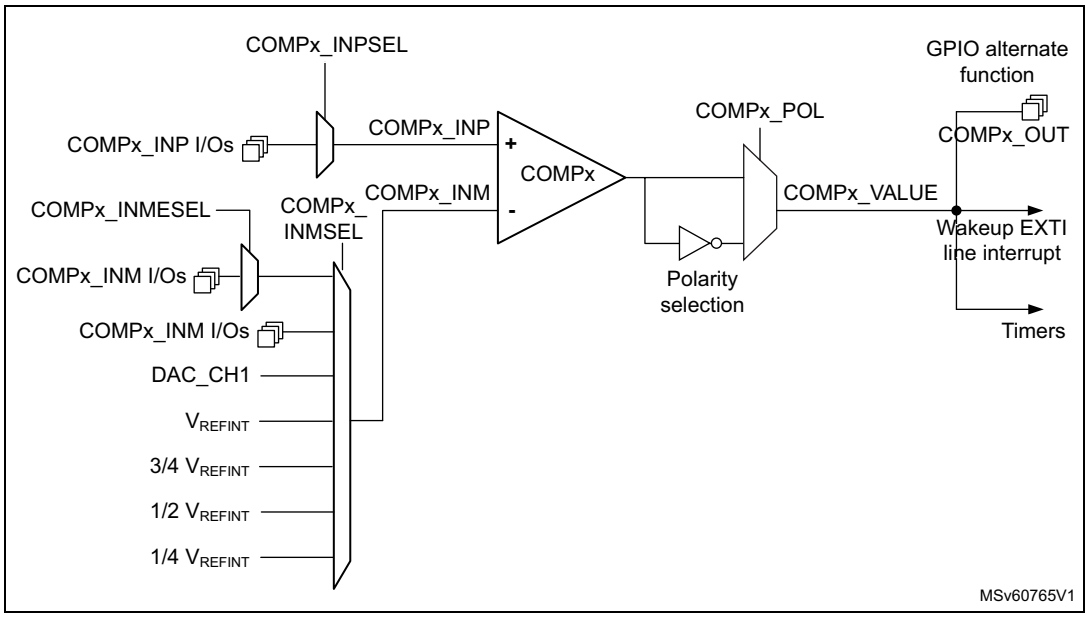
- Configurable plus and minus inputs used for flexible voltage:
  - Multiplexed I/O pins
  - DAC channel1
  - Internal reference voltage and three sub multiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed / consumption
- Redirection of outputs to an I/O or to timer inputs for triggering:
  - break events for fast PWM shutdowns
- Blanking of comparator outputs
- Window comparator
- Interrupt generation capability with wake up from Sleep and Stop modes (through the EXTI controller)

19.3 COMP functional description

19.3.1 COMP block diagram

The block diagram of the comparators is shown in the figure below.

Figure 80. Comparator block diagram



19.3.2 COMP pins and internal signals

The I/Os used as comparator inputs must be configured in analog mode in the GPIO registers.

The comparator outputs can be connected to the I/Os through their alternate functions (refer to the product datasheet).

The outputs can also be internally redirected to a variety of timer inputs for the following purposes:

- emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- cycle-by-cycle current control, using OCREF_CLR inputs
- input capture for timing measurements

The comparator output can be simultaneously redirected internally and externally.

Table 110. COMP1 input plus assignment

COMP1_INP	COMP1_INPSEL
PB4	00
PB2	01

Table 111. COMP1 input minus assignment

COMP1_INM	COMP1_INMSEL[2:0]	COMP1_INMESEL[1:0]
1/4 V_{REFINT}	000	Not affected
1/2 V_{REFINT}	001	Not affected
3/4 V_{REFINT}	010	Not affected
V_{REFINT}	011	Not affected
DAC channel1	100	Not affected
Reserved	101	Not affected
PB3	110	Not affected
PA10	111	00
PA11	111	01
PA15	111	10
Reserved	111	11

Table 112. COMP2 input plus assignment

COMP2_INP	COMP2_INPSEL
PB4	00
PB1	01
PA15	10

Table 113. COMP2 input minus assignment

COMP2_INM	COMP2_INMSEL[2:0]	COMP2_INMESEL[1:0]
1/4 V_{REFINT}	000	Not affected
1/2 V_{REFINT}	001	Not affected
3/4 V_{REFINT}	010	Not affected
V_{REFINT}	011	Not affected
DAC channel1	100	Not affected
Reserved	101	Not affected
PB3	110	Not affected
PB2	111	00
PA10	111	01
PA11	111	10
Reserved	111	11

19.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the APB2 clock.

There is no clock enable control bit provided in the RCC controller. Reset and clock enable bits are common for COMP and SYSCFG.

Note: ***Important:*** *The polarity selection logic and the output redirection to the port works independently from the APB2 clock. This allows the comparator to work even in Stop mode.*

19.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications with specific functional safety requirements, the comparator configuration can be protected against undesired alteration that may happen, for example, at program counter corruption.

For this purpose, the comparator configuration registers can be write-protected (read-only).

Once the programming is completed, the COMPx LOCK bit can be set to 1. This causes the whole register to become read-only, including the COMPx LOCK bit.

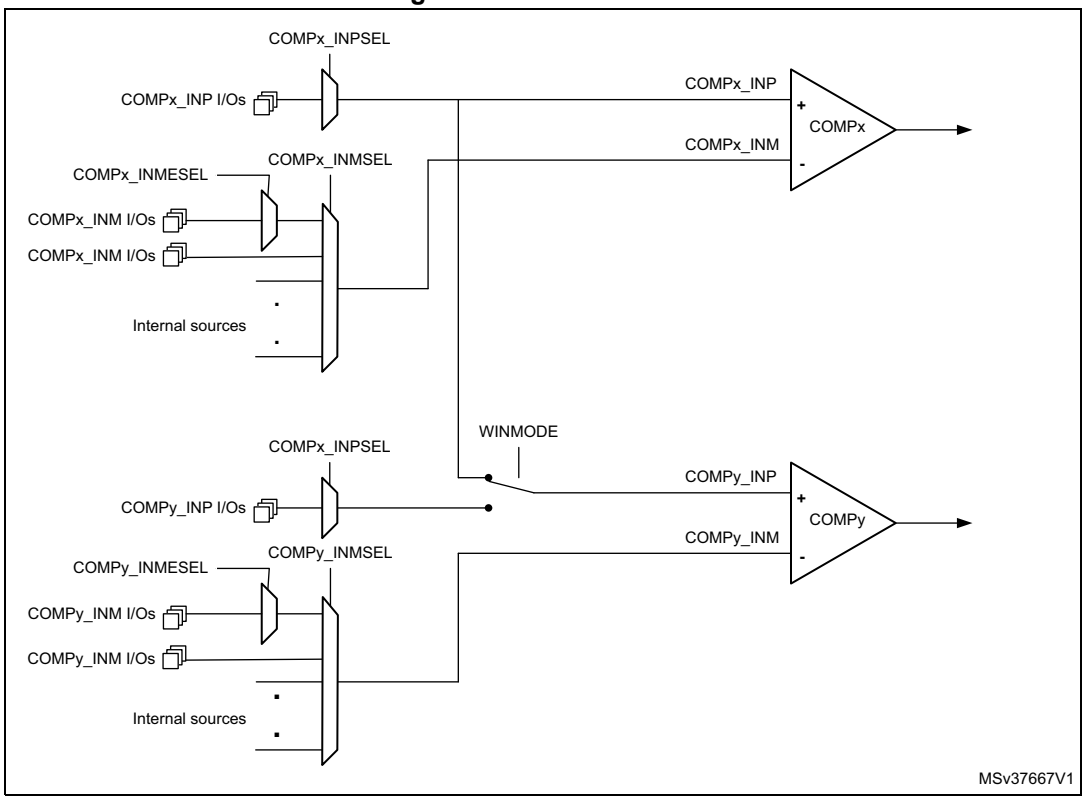
The write protection can only be removed through the MCU reset.

19.3.5 Window comparator

The purpose of the window comparator is to monitor the analog voltage and check that it is comprised within the specified voltage range defined by lower and upper thresholds.

COMP1 and COMP2 can be utilized to create window comparator. The monitored analog voltage is connected to the non-inverting (plus) inputs of comparators connected together, and the upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators. The two non-inverting inputs can be connected internally together by enabling the WINMODE bit to save one I/O for other purposes.

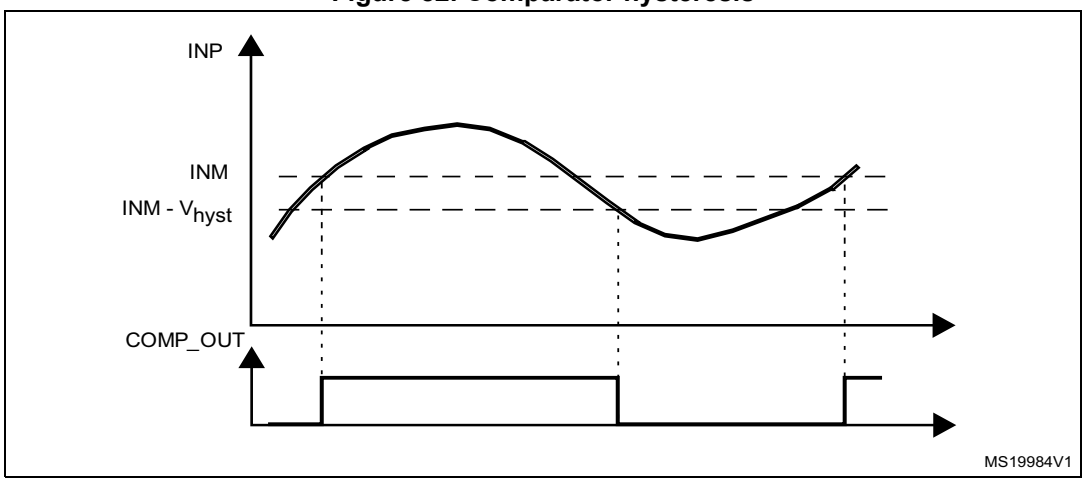
Figure 81. Window mode



19.3.6 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

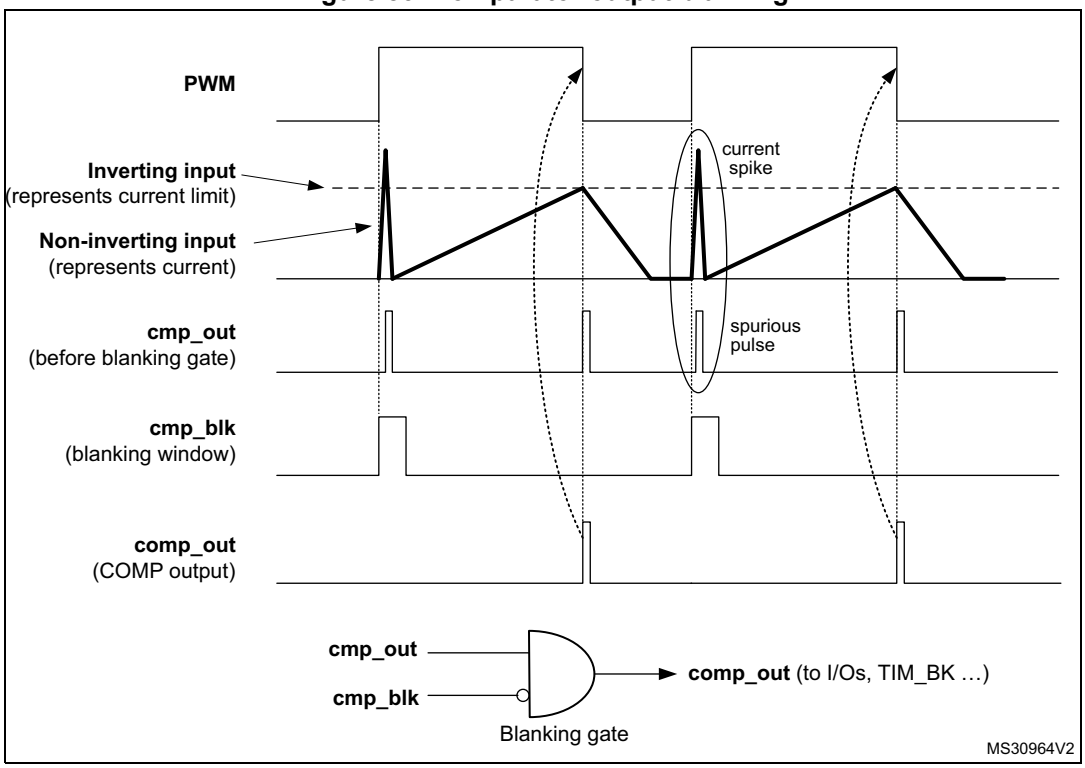
Figure 82. Comparator hysteresis



19.3.7 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). It uses a blanking window defined with a timer output compare signal. Refer to the register description for selectable blanking signals. The blanking signal gates the internal comparator output such as to clean the comp_out from spurious pulses due to current spikes, as depicted in the figure below.

Figure 83. Comparator output blanking



19.3.8 COMP power and speed modes

COMP1 and COMP2 power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

PWRMODE[1:0] bits in COMPx_CSR registers can be programmed as follows:

00: High speed/full power

01: Medium speed/medium power

10: Medium speed/medium power

11: Very-low speed/ultra-low-power

19.4 COMP low-power modes

Table 114. Comparator behavior in the low-power modes

Mode	Description
Sleep	No effect on the comparators Comparator interrupts cause the device to exit the Sleep mode.
LPRun	No effect
LPSleep	No effect on the comparators Comparator interrupts cause the device to exit the LPSleep mode.
Stop 0	No effect on the comparators Comparator interrupts cause the device to exit the Stop mode.
Stop 1	
Stop 2	
Standby	COMP registers are powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

19.5 COMP interrupts

The comparator outputs are internally connected to the extended interrupts and events controller (EXTI). Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from low-power modes.

Refer to the “Interrupt and events” section for more details.

The following sequence enables the COMPx interrupt through EXTI block:

1. Configure and enable the EXTI line corresponding to the COMPx output event in interrupt mode and select the rising, falling or both edges sensitivity.
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines.
3. Enable the COMPx.

Table 115. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop modes	Exit from Standby mode
COMP1 output	VALUE in COMP1_CSR	Through EXTI	Yes	Yes	Not applicable
COMP2 output	VALUE in COMP2_CSR	Through EXTI	Yes	Yes	Not applicable

19.6 COMP registers

19.6.1 COMP1 control and status register (COMP1_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	INMESEL[1:0]		Res.	SCALEN	BRGEN	Res.	BLANKING[2:0]			HYST[1:0]	
rs	r				rw	rw		rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL[1:0]		INMSEL[2:0]			PWRMODE[1:0]		Res.	EN
rw							rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **LOCK**: locks the whole content of the register, COMP1_CSR[31:0]
This bit is set by software and cleared by a hardware system reset.
0: COMP1_CSR[31:0] are read/write.
1: COMP1_CSR[31:0] are read-only.

Bit 30 **VALUE**: COMP1 output status bit
This bit is read-only. It reflects the current COMP1 output taking into account the effect of the POLARITY bit.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:25 **INMESEL[1:0]**: COMP1 input minus extended selection
These bits are set and cleared by software. They select which extended GPIO input is connected to the input minus of COMP1, if INMSEL[2:0] = 111.
00: PA10
01: PA11
10: PA15
11: reserved

Bit 24 Reserved, must be kept at reset value.

Bit 23 **SCALEN**: voltage scaler enable
This bit is set and cleared by software. It enables outputs of the V_{REFINT} divider available on the minus input of COMP1.
0: Bandgap scaler disabled (if SCALEN bit of COMP2_CSR register is also reset)
1: Bandgap scaler enabled

Bit 22 **BRGEN**: scaler bridge enable
This bit is set and cleared by software. It enables the bridge of the scaler.
If SCALEN is set and BRGEN is reset, the BG voltage reference is available but not 1/4 BGAP, 1/2 BGAP or 3/4 BGAP. The BGAP value is sent instead of 1/4 BGAP, 1/2 BGAP, 3/4 BGAP. If SCALEN and BRGEN are both set, 1/4 BGAP 1/2 BGAP 3/4 BGAP and BGAP voltage references are available.
0: Scaler resistor bridge disabled (if BRGEN bit of COMP2_CSR register is also reset)
1: Scaler resistor bridge enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:18 **BLANKING[2:0]**: COMP1 blanking source selection
These bits select which timer output controls the COMP1 output blanking.
000: No blanking
001: TIM1 OC5 selected as blanking source
010: TIM2 OC3 selected as blanking source
Others: reserved

Bits 17:16 **HYST[1:0]**: COMP1 hysteresis selection
These bits are set and cleared by software. They select the COMP1 hysteresis voltage.
00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 15 **POLARITY**: COMP1 polarity selection
This bit is set and cleared by software. It inverts COMP1 polarity.
0: COMP1 output value not inverted
1: COMP1 output value inverted

Bits 14:9 Reserved, must be kept at reset value.

Bits 8:7 **INPSEL[1:0]**: COMP1 input plus selection
These bits are set and cleared by software.
00: PB4
01: PB2
10: reserved
11: reserved

Bits 6:4 **INMSEL[2:0]**: COMP1 input minus selection
These bits are set and cleared by software. They select which input is connected to the input minus of COMP1.
000: 1/4 V_{REFINT}
001: 1/2 V_{REFINT}
000: 3/4 V_{REFINT}
011: V_{REFINT}
100: DAC channel1
101: reserved
110: PB3
111: GPIOx selected by INMESEL[1:0] bits

Bits 3:2 **PWRMODE[1:0]**: COMP1 power mode
These bits are set and cleared by software. They control the power and speed of COMP1.
00: High speed
01: Medium speed
10: Medium speed
11: Ultra low-power

Bit 1 Reserved, must be kept at reset value.

Bit 0 **EN**: COMP1 enable
This bit is set and cleared by software. It switches COMP1 on.
0: COMP1 switched off
1: COMP1 switched on

19.6.2 COMP2 control and status register (COMP2_CSR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	INMESEL[1:0]		Res.	SCAL EN	BRGEN	Res.	BLANKING[2:0]			HYST[1:0]	
rs	r				rw	rw		rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLA RITY	Res.	Res.	Res.	Res.	Res.	WIN MODE	INPSEL[1:0]		INMSEL[2:0]			PWRMODE[1:0]		Res.	EN
rw						rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **LOCK**: locks the whole content of the register, COMP2_CSR[31:0]
This bit is set by software and cleared by a hardware system reset.
0: COMP2_CSR[31:0] are read/write.
1: COMP2_CSR[31:0] are read-only.

Bit 30 **VALUE**: COMP2 output status bit
This bit is read-only. It reflects the current COMP2 output taking into account the effect of the POLARITY bit.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:25 **INMESEL[1:0]**: COMP2 input minus extended selection
These bits are set and cleared by software. They select which extended GPIO input is connected to the input minus of COMP2, if INMSEL[2:0] = 111.
00: PB2
01: PA10
10: PA11
11: reserved

Bit 24 Reserved, must be kept at reset value.

Bit 23 **SCALEN**: voltage scaler enable
This bit is set and cleared by software. It enables outputs of the V_{REFINT} divider available on the minus input of COMP2.
0: Bandgap scaler disabled (if SCALEN bit of COMP1_CSR register is also reset)
1: Bandgap scaler enabled

Bit 22 **BRGEN**: scaler bridge enable
This bit is set and cleared by software. It enables the bridge of the scaler.
If SCALEN is set and BRGEN is reset, the BG voltage reference is available but not 1/4 BGAP, 1/2 BGAP or 3/4 BGAP. The BGAP value is sent instead of 1/4 BGAP, 1/2 BGAP, 3/4 BGAP. If SCALEN and BRGEN are both set, 1/4 BGAP 1/2 BGAP 3/4 BGAP and BGAP voltage references are available.
0: Scaler resistor bridge disabled (if BRGEN bit of COMP1_CSR register is also reset)
1: Scaler resistor bridge enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:18 **BLANKING[2:0]**: COMP2 blanking source selection
These bits select which timer output controls the COMP2 output blanking.
000: No blanking
001: TIM1 OC5 selected as blanking source
010: TIM2 OC3 selected as blanking source
Others: reserved

Bits 17:16 **HYST[1:0]**: COMP2 hysteresis selection
These bits are set and cleared by software. They select the COMP2 hysteresis voltage.
00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 15 **POLARITY**: COMP2 polarity selection
This bit is set and cleared by software. It inverts COMP2 polarity.
0: COMP2 output value not inverted
1: COMP2 output value inverted

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **WINMODE**: window mode selection
This bit is set and cleared by software. It selects the window mode of the comparators. If set, both positive inputs of comparators are connected together.
0: COMP2 input plus is not connected to COMP1.
1: COMP2 input plus is connected to COMP1.

Bits 8:7 **INPSEL[1:0]**: COMP2 input plus selection
These bits are set and cleared by software.
00: PB4
01: PB1
10: PA15
11: reserved

Bits 6:4 **INMSEL[2:0]**: COMP2 input minus selection
These bits are set and cleared by software. They select which input is connected to the input minus of COMP2.
000: 1/4 V_{REFINT}
001: 1/2 V_{REFINT}
000: 3/4 V_{REFINT}
011: V_{REFINT}
100: DAC channel1
101: reserved
110: PB3
111: GPIOx selected by INMESEL[1:0] bits

Bits 3:2 **PWRMODE[1:0]**: COMP2 power mode
These bits are set and cleared by software. They control the power and speed of COMP2.
00: High speed
01: Medium speed
10: Medium speed
11: Ultra low-power

Bit 1 Reserved, must be kept at reset value.

Bit 0 **EN**: COMP2 enable

This bit is set and cleared by software. It switches COMP2 on.

0: COMP2 switched off

1: COMP2 switched on

19.6.3 COMP register map

Table 116. COMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	COMP1_CSR	LOCK	VALUE	Res.	Res.	Res.	INMESEL[1:0]		Res.	SCALEN	BRGEN	Res.	BLANKING[2:0]			HYST[1:0]		POLARITY	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL[1:0]		INMSEL[2:0]			PWRMODE[1:0]		Res.	EN
	Reset value	0	0				0	0		0	0		0	0	0	0	0	0							0	0	0	0	0	0	0		0
0x04	COMP2_CSR	LOCK	VALUE	Res.	Res.	Res.	INMESEL[1:0]		Res.	SCALEN	BRGEN	Res.	BLANKING[2:0]			HYST[1:0]		POLARITY	Res.	Res.	Res.	Res.	Res.	WINMODE	INPSEL[1:0]		INMSEL[2:0]			PWRMODE[1:0]		Res.	EN
	Reset value	0	0				0	0		0	0		0	0	0	0	0	0						0	0	0	0	0	0	0	0		0

Refer to *Section 2.4* for the register boundary addresses.

20 True random number generator (RNG)

20.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is a NIST SP 800-90B compliant entropy source that can be used to construct a non-deterministic random bit generator (NDRBG).

The RNG true random number generator has been pre-certified NIST SP800-90B. It has also been tested using German BSI statistical tests of AIS-31 (T0 to T8).

20.2 RNG main features

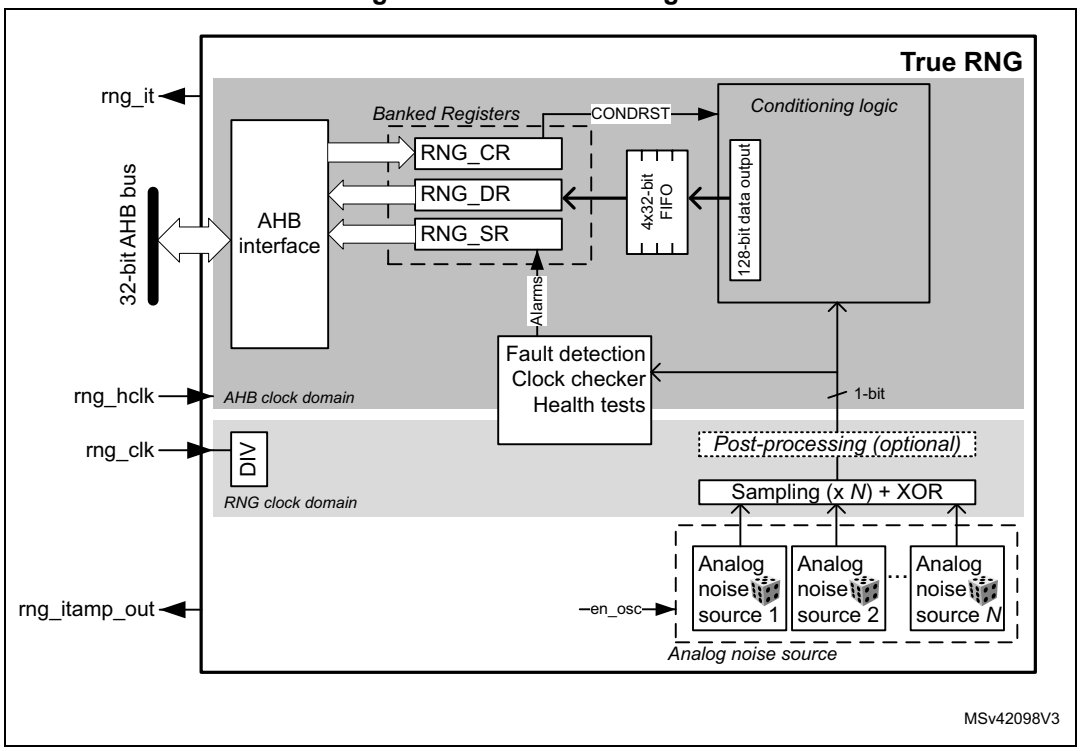
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as entropy source to construct a non-deterministic random bit generator (NDRBG).
- In the NIST configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- It embeds start-up and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- It has an AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

20.3 RNG functional description

20.3.1 RNG block diagram

Figure 84 shows the RNG block diagram.

Figure 84. RNG block diagram



20.3.2 RNG internal signals

Table 117 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 117. RNG internal input/output signals

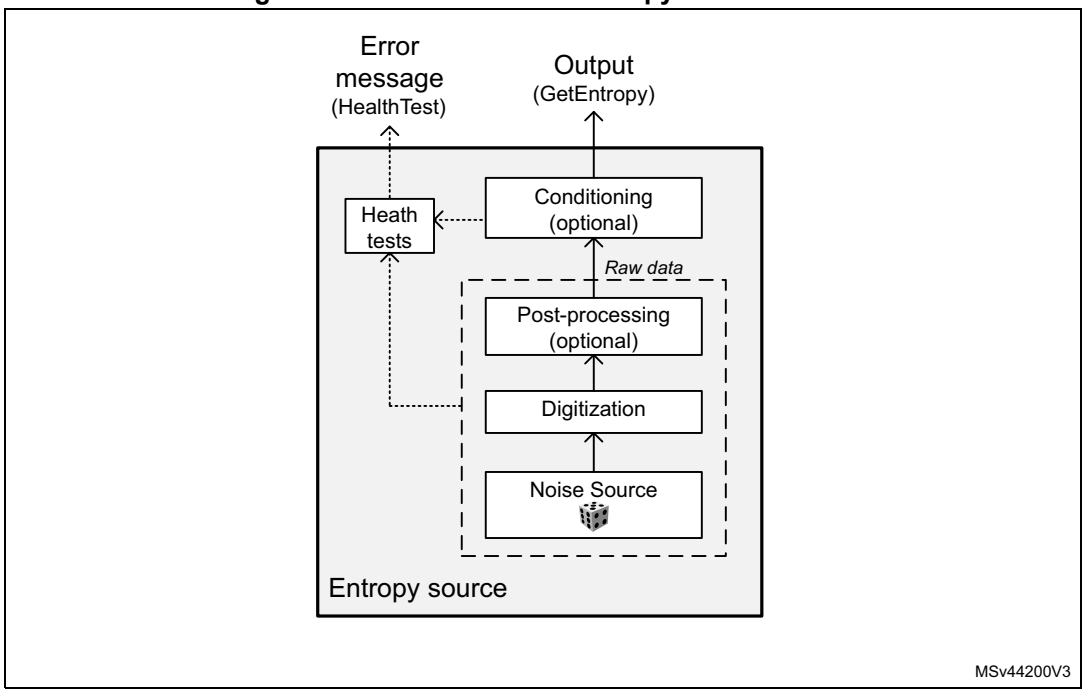
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk

20.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary RNG integrates all the required NIST components depicted on *Figure 85*. Those components are an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

Figure 85. NIST SP800-90B entropy source model



The components pictured above are detailed hereafter.

Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x6), each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 20.3.8: RNG low-power usage*.
- The XORing of all the noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

Note: In Section 20.6: RNG entropy source validation recommended RNG clock frequencies and associated divider value are given.

Post processing

In NIST configuration no post-processing is applied to sampled noise source. In non-NIST configuration B (as defined in *Section 20.6.2*) a normalization debiasing is applied, that is half of the bits are taken from the sampled noise source, half of the bits are taken from inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in *Section 20.5: RNG processing time*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 20.5: RNG processing time*.

Whenever a random number is available through the RNG_DR register the DRDY flag transitions from 0 to 1. This flag remains high until output buffer becomes empty after reading four words from the RNG_DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (configuration A in *Section 20.6.2*).

1. Start-up health tests, performed after reset and before the first use of the RNG as entropy source
   - Repetition count test, flagging an error when the noise source has provided more than 42 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 628 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source
   - Repetition count test, similar to the one running in start-up tests.
   - Adaptive proportion test, similar to the one running in start-up tests.
3. Vendor specific continuous tests
   - Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrences of 2-bit patterns (01 or 10).
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than AHB clock cycle divided by 32.
4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and re-running the startup tests (see software reset sequence in *Section 20.3.4: RNG initialization*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 20.3.7: Error management*.

Note: An interrupt can be generated when an error is detected.

Above health test thresholds are modified by changing value in RNG_HTCR register. See Section 20.6: RNG entropy source validation for details.

20.3.4 RNG initialization

The RNG simplified state machine is pictured on *Figure 86*.

After enabling the RNG (RNGEN = 1 in RNG_CR) the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 cycles of RNG clock cycles (before divider) before starting to sample analog output and filling 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering start-up behavior test on the raw data samples and known-answer tests.
3. When start-up health tests are completed. During this time three 128-bit noise source samples are used.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 20.5: RNG processing time*.

Figure 86. RNG initialization overview

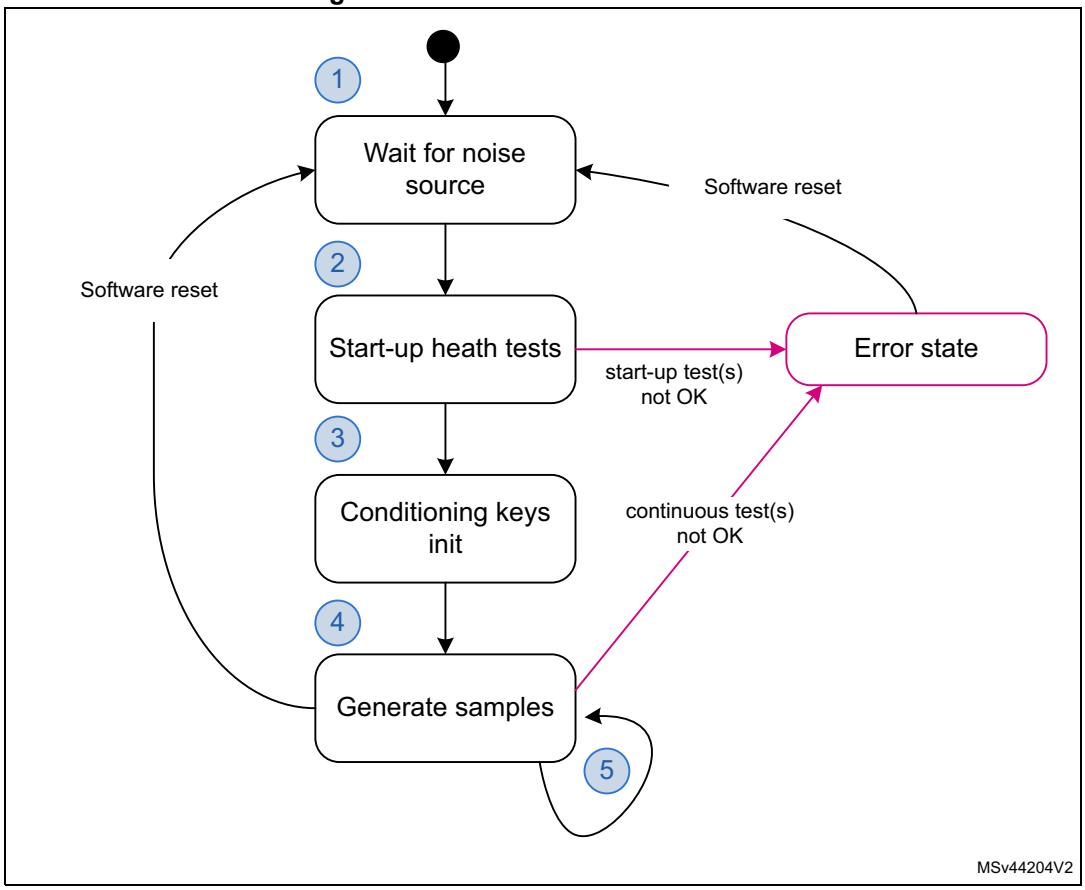


Figure 86 also highlights a possible software reset sequence, implemented by:

1. Writing bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
2. Then writing RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
3. Wait for random number to be ready, after initialization completes

Note: When RNG peripheral is reset through RCC (hardware reset), the RNG configuration for optimal randomness is lost in RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

20.3.5 RNG operation

Normal operations

To run the RNG using interrupts, the following steps are recommended:

1. Consult *Section 20.6: RNG entropy source validation* and verify if a specific RNG configuration is required for the application.
   - If it is the case, write in the RNG_CR register the bit CONDRST = 1 together with the correct RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0, the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.
   - If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in *Section 20.3.7* must be used.

To run the RNG in polling mode following steps are recommended:

1. Consult *Section 20.6: RNG entropy source validation* and verify if a specific RNG configuration is required for the application.
   - If it is the case write in the RNG_CR register the bit CONDRST = 1 together with the correction RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If it is not the case only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four

additional words can be read by the application (in this case the DRDY bit is still high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in *Section 20.3.7* must be used.

Note: When data is not ready (DRDY = 0) RNG_DR returns zero.
It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required it is possible to select a faster RNG configuration by using the RNG configuration “B”, described in *Section 20.6: RNG entropy source validation*. The gain in random number generation speed is summarized in *Section 20.5: RNG processing time*.

Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 20.3.8: RNG low-power usage*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 20.3.3: Random number generation*.

20.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 20.6: RNG entropy source validation*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before internal divider **must be higher** than AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 20.3.1: RNG block diagram* for details (AHB and RNG clock domains).

20.3.7 Error management

In parallel to random number generation an health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 20.3.6: RNG clocking*) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when clocking condition is normal.

Note: The clock error has no impact on generated random numbers, that is the application can still read RNG_DR register.

CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy.

The following sequence must be used to fully recover from a seed error:

1. Software reset by writing CONDRST at 1 and at 0 (see bitfield description for details).
2. wait for CONDRST to be cleared in the RNG_CR register, then confirm that SEIS is cleared in the RNG_SR register.
3. wait for SECS to be cleared by RNG. The random number generation is now back to normal.

20.3.8 RNG low-power usage

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode) and a complete conditioning round is done.
  Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 20.5: RNG processing time on page 537*.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1, unless the application resets the conditioning logic using CONDRST bit in the RNG_CR register.

20.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 20.3.7: Error management*
- Clock error, see *Section 20.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 118*.

Table 118. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write 0 to SEIS or write CONDRST to 1 then to 0
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

20.5 RNG processing time

In recommended configuration A described in *Table 119*, the time between two sets of four 32-bit data is either:

- 206 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

With $f_{threshold} = 1.6 \times f_{RNG}$, for instance 77 MHz if f_{RNG}= 48 MHz.Value N is 2.

Note: When CLKDIV is different from zero, f_{RNG} must take into account the internal divider ratio.

If configuration B is selected the performance figures become:

- 206 AHB cycles if $f_{AHB} < f_{threshold}$ or
- 32 RNG cycles $f_{AHB} \geq f_{threshold}$

with $f_{threshold} = 6.5 \times f_{RNG}$.

20.6 RNG entropy source validation

20.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite. The results can be provided on demand or the customer can reproduce the tests.

20.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk= 48 MHz
- RNG configurations described in *Table 119: RNG configurations*. Note that only configuration A can be certified NIST SP800-90B.

Table 119. RNG configurations

RNG Config.	RNG_CR bits						Loop number (N)	RNG_ HTCR register[1]
	NISTC bit	RNG_ CONFIG1 [5:0]	CLKDIV [3:0]	RNG_ CONFIG2 [2:0]	RNG_ CONFIG3 [3:0]	CED bit		
A	0	0x0F	0x0	0x0	0xD	0	2	0x0000 AA74[2]
B	1	0x18	0x0	0x0	0x0	0	1	0x0000 AA74

1. When writing this register magic number 0x17590ABC must be written immediately before the indicated value
2. Corresponds to 42 for repetition tests and 628 for adaptive tests. See *Health checks on page 532* for details

20.6.3 Data collection

In order to run statistical tests it is required to collect samples from the entropy source at raw data level as well as at the output of the entropy source. For details on data collection and the running of statistical test suites refer to “STM32 microcontrollers random number generation validation using NIST statistical test suite” application note (AN4230) available from *www.st.com*.

Contact STMicroelectronics if above samples need to be retrieved for the product.

20.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

20.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	Res.	Res.	RNG_CONFIG1[5:0]						CLKDIV[3:0]			
rs	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw			rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG Config lock

0: Writes to the RNG_HTCR and RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_HTCR and RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

This bitfield is set once: if this bit is set it can only be reset to 0 if RNG is reset.

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_CR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that set any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

When CONDRST is set to 0 by software its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:20 **RNG_CONFIG1[5:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 20.6: RNG entropy source validation*.

Writing any bit of RNG_CONFIG1 is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing these bits is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Refer to RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration per 128-bit output two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 20.6: RNG entropy source validation* for proposed configuration.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection is enable

1: Clock error detection is disable

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED the RNG must be disabled.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt Enable

0: RNG Interrupt is disabled

1: RNG Interrupt is enabled. An interrupt is pending as soon as DRDY = 1, SEIS = 1 or CEIS = 1 in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.

1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.

20.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct (fRNGCLK> fHCLK/32)

1: The RNG clock before internal divider is detected too slow (fRNGCLK< fHCLK/32)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

- Run-time repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0 or 1, or more than 32 consecutive occurrence of two bits patterns 01 or 10)
- Start-up or continuous adaptive proportion test on noise source failed.
- Start-up post-processing/conditioning sanity check failed.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct (fRNGCLK> fHCLK/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow (fRNGCLK< fHCLK/32).

Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data Ready

0: The RNG_DR register is not yet valid, no random data is available.

1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY = 1.

20.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when DRDY = 1 and value is not 0x0, even if RNGEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data which are valid when DRDY = 1. When DRDY = 0 RNDATA value is zero.

When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

20.7.4 RNG health test control register (RNG_HTCR)

Address offset: 0x010

Reset value: 0x0000 5A4E

Writing in RNG_HTCR is taken into account only if CONDRST bit is set, and CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK=1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTCFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTCFG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests. See *Section 20.6: RNG entropy source validation* for the recommended value.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

When reading or writing this register magic number; 0x17590ABC must be written immediately before to RNG_HTCR register.

20.7.5 RNG register map

Table 120. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	Res.	Res.	RNG_CONFIG1[5:0]						CLKDIV[3:0]				RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0					0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
	Reset value																										0	0			0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	RNG_HTCR	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	0	1	0	0	1	0	0	1	1	1	0

Refer to *Section 2.4* for the register boundary addresses.

21 AES hardware accelerator (AES)

21.1 Introduction

The AES hardware accelerator (AES) encrypts or decrypts data, using an algorithm and implementation fully compliant with the advanced encryption standard (AES) defined in Federal information processing standards (FIPS) publication 197.

The peripheral supports CTR, GCM, GMAC, CCM, ECB, and CBC chaining modes for key sizes of 128 or 256 bits.

AES is an AMBA AHB slave peripheral accessible through 32-bit single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels required).

21.2 AES main features

- Compliance with NIST *"Advanced encryption standard (AES)*, *FIPS publication 197*" from November 2001
- 128-bit data block processing
- Support for cipher key lengths of 128-bit and 256-bit
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block of data with, respectively, 128-bit or 256-bit key
- Integrated round key scheduler to compute the last round key for ECB/CBC decryption
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only
- 256-bit write-only register for storing the cryptographic key (eight 32-bit registers)
- 128-bit register for storing initialization vector (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control with support of single-transfer direct memory access (DMA) using two channels (one for incoming data, one for processed data)
- Data-swapping logic to support 1-, 8-, 16- or 32-bit data
- Possibility for software to suspend a message if AES needs to process another message with a higher priority, then resume the original message

21.3 AES implementation

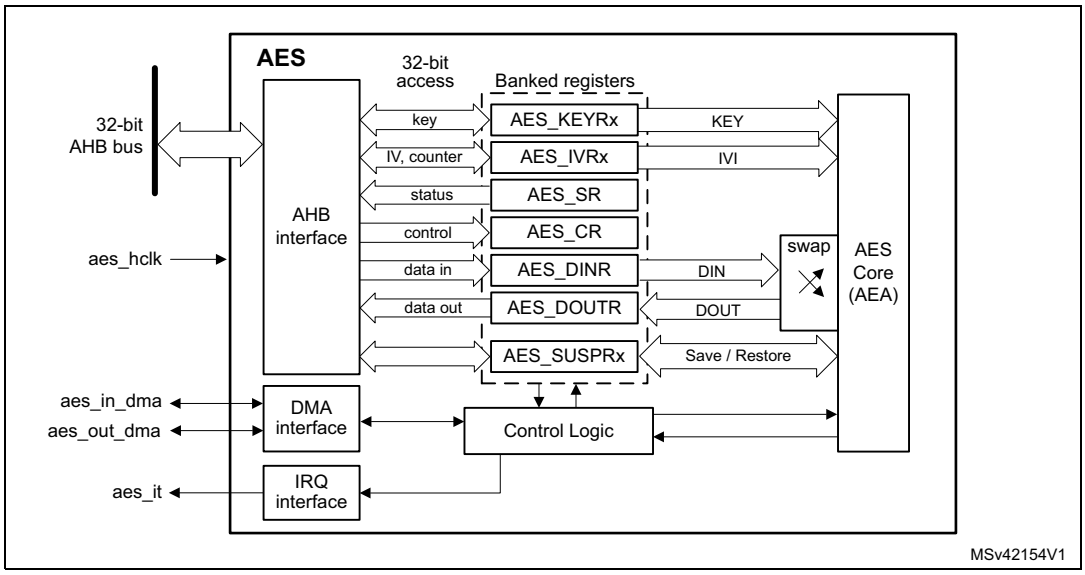
The devices have one AES peripheral.

21.4 AES functional description

21.4.1 AES block diagram

Figure 87 shows the block diagram of AES.

Figure 87. AES block diagram



21.4.2 AES internal signals

Table 121 describes the user relevant internal signals interfacing the AES peripheral.

Table 121. AES internal input/output signals

Signal name	Signal type	Description
aes_hclk	Input	AHB bus clock
aes_it	Output	AES interrupt request
aes_in_dma	Input/Output	Input DMA single request/acknowledge
aes_out_dma	Input/Output	Output DMA single request/acknowledge

21.4.3 AES cryptographic core

Overview

The AES cryptographic core consists of the following components:

- AES core algorithm (AEA)
- multiplier over a binary Galois field (GF2mul)
- key input
- initialization vector (IV) input
- chaining algorithm logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks (four words) with 128-bit or 256-bit key length. Depending on the chaining mode, the AES requires zero or one 128-bit initialization vector IV.

The AES features the following modes of operation:

- **Mode 1:**

  Plaintext encryption using a key stored in the AES_KEYRx registers
- **Mode 2:**

  ECB or CBC decryption key preparation. It must be used prior to selecting Mode 3 with ECB or CBC chaining modes. The key prepared for decryption is stored automatically in the AES_KEYRx registers. Now the AES peripheral is ready to switch to Mode 3 for executing data decryption.
- **Mode 3:**

  Ciphertext decryption using a key stored in the AES_KEYRx registers. When ECB and CBC chaining modes are selected, the key must be prepared beforehand, through Mode 2.

Note: Mode 2 is only used when performing ECB and CBC decryption.

The operating mode is selected by programming the MODE[1:0] bitfield of the AES_CR register. It may be done only when the AES peripheral is disabled.

Typical data processing

Typical usage of the AES is described in *Section 21.4.4: AES procedure to perform a cipher operation on page 551*.

Note: The outputs of the intermediate AEA stages are never revealed outside the cryptographic boundary, with the exclusion of the IVI bitfield.

Chaining modes

The following chaining modes are supported by AES, selected through the CHMOD[2:0] bitfield of the AES_CR register:

- Electronic code book (ECB)
- Cipher block chaining (CBC)
- Counter (CTR)
- Galois counter mode (GCM)
- Galois message authentication code (GMAC)
- Counter with CBC-MAC (CCM)

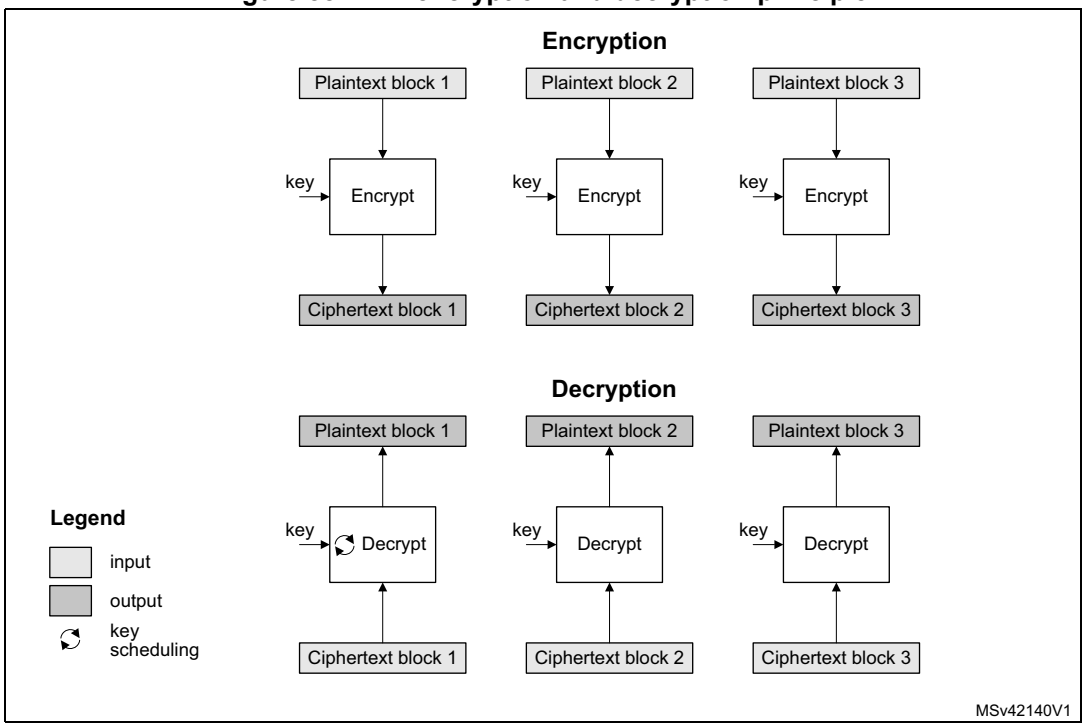
Note: The chaining mode may be changed only when AES is disabled (bit EN of the AES_CR register cleared).

Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from *Section 21.4.8: AES basic chaining modes (ECB, CBC)*.

Electronic codebook (ECB) mode

Figure 88. ECB encryption and decryption principle

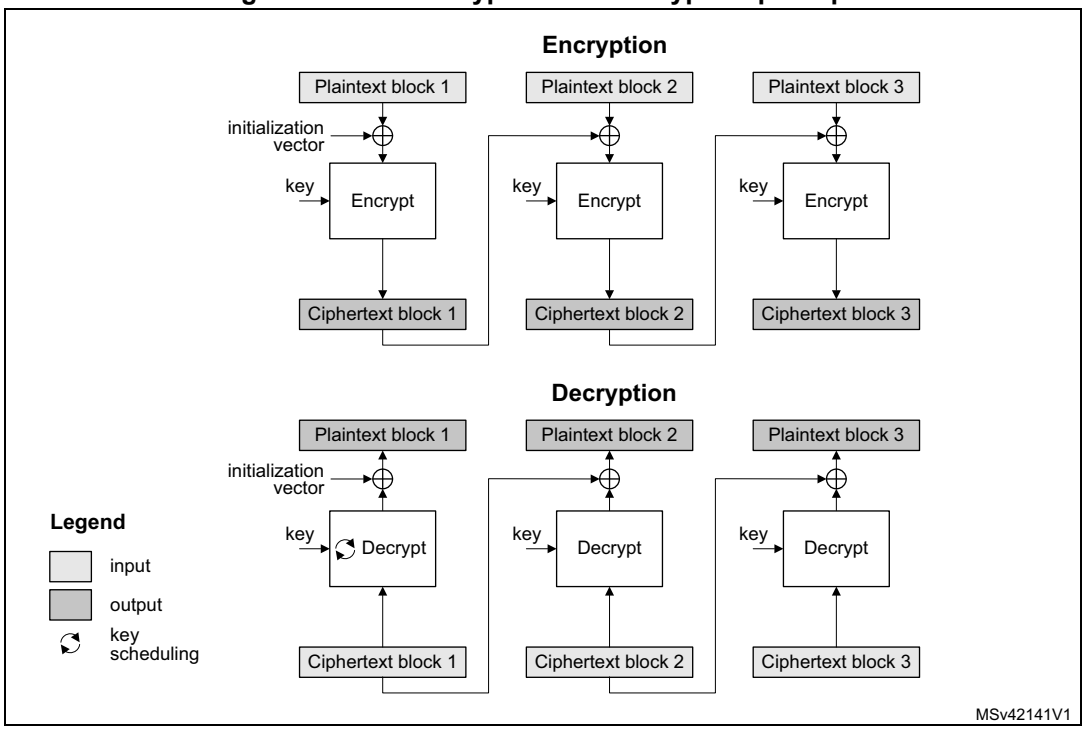


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.

Cipher block chaining (CBC) mode

Figure 89. CBC encryption and decryption principle

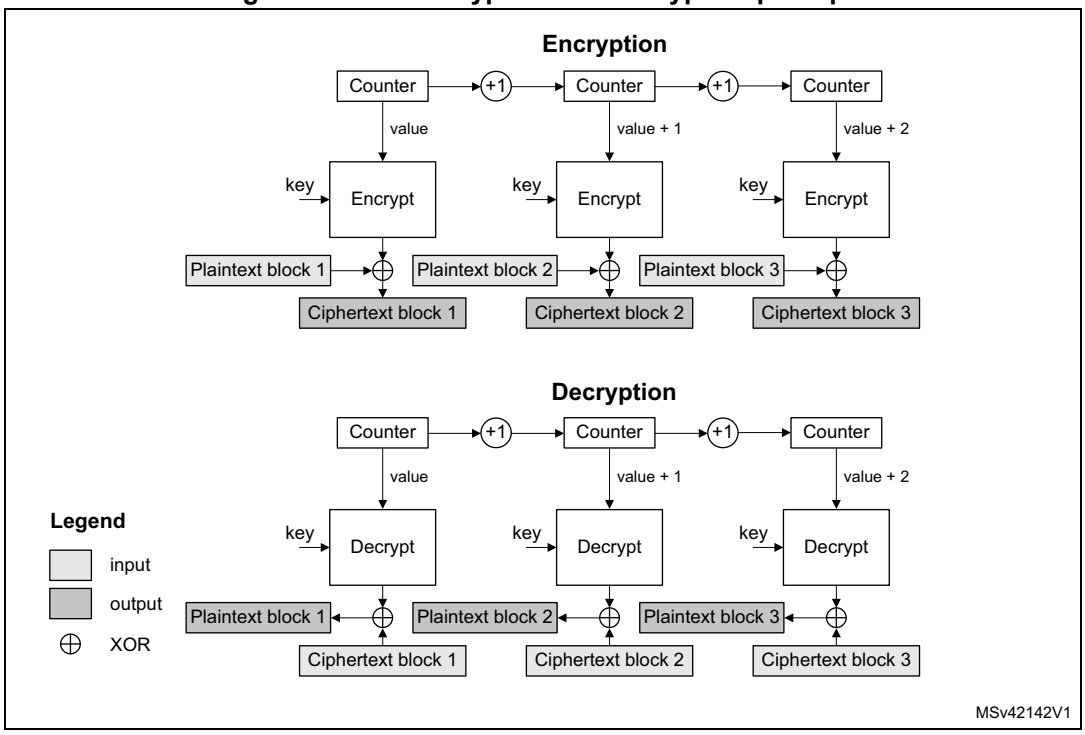


In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

Note: For decryption, a special key scheduling is required before processing the first block.

Counter (CTR) mode

Figure 90. CTR encryption and decryption principle

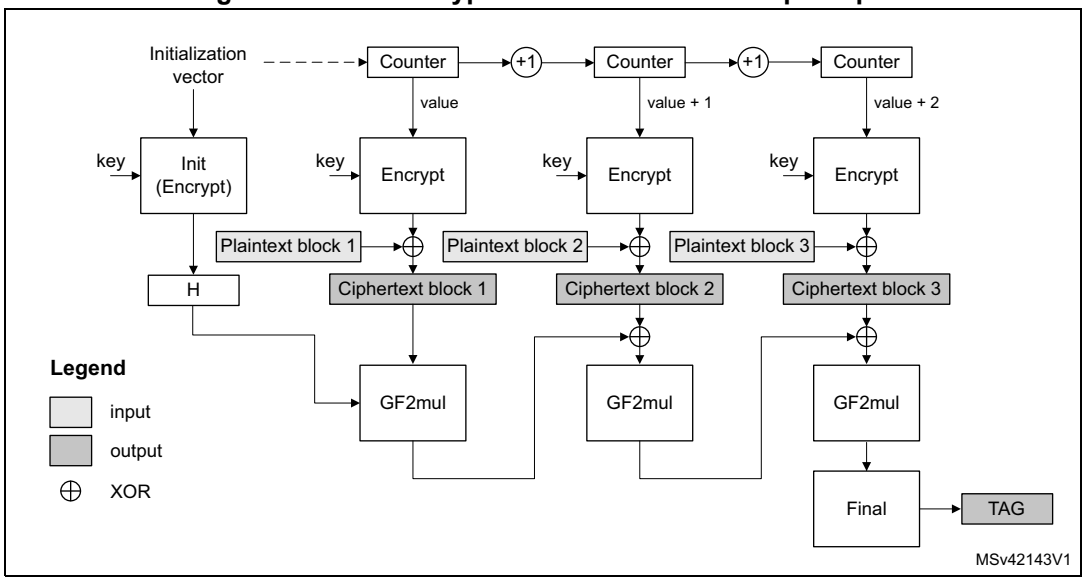


The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Note: Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption, since in this chaining scheme the AES core is always used in encryption mode for producing the key stream, or counter blocks.

Galois/counter mode (GCM)

Figure 91. GCM encryption and authentication principle

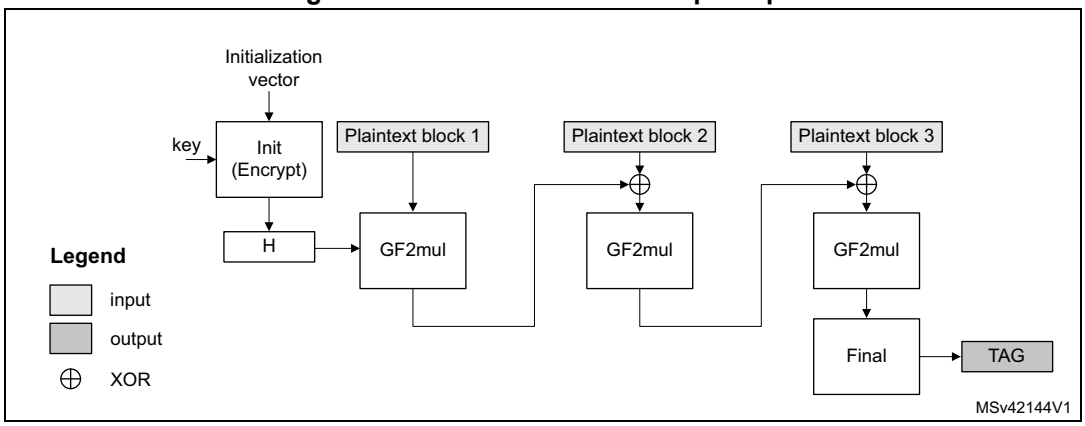


In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. It requires an initial value and a particular 128-bit block at the end of the message.

Galois message authentication code (GMAC) principle

Figure 92. GMAC authentication principle

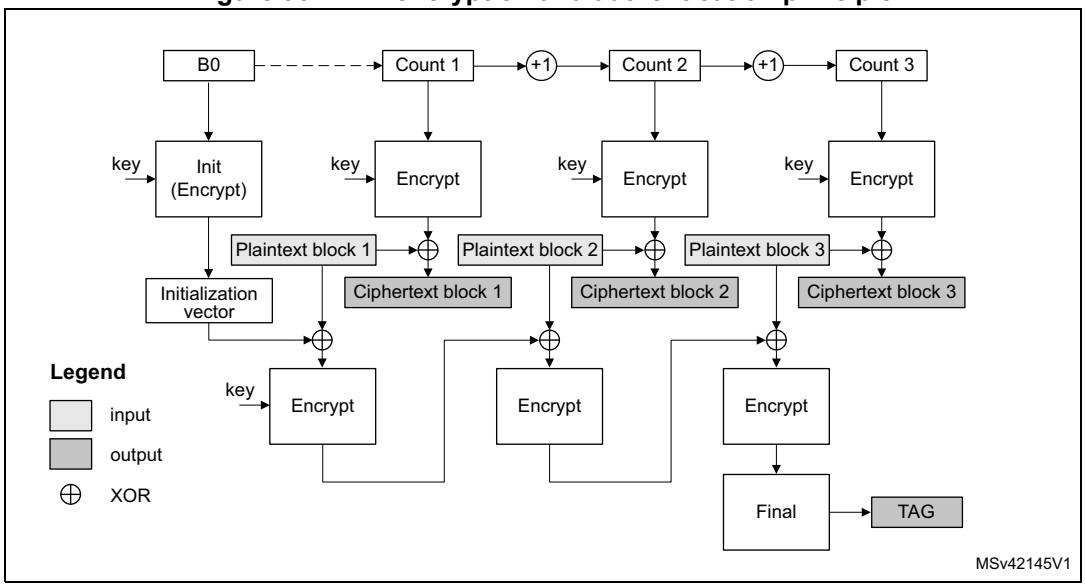


Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload).

Counter with CBC-MAC (CCM) principle

Figure 93. CCM encryption and authentication principle



In Counter with cipher block chaining-message authentication code (CCM) mode, the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and the corresponding MAC (also known as tag). It is described by NIST in *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

CCM mode is based on AES in counter mode for confidentiality and it uses CBC for computing the message authentication code. It requires an initial value.

Like GCM, the CCM chaining mode can be applied on a message composed only by plaintext authenticated data (that is, only header, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM/GMAC), and its use is not recommended by NIST.

21.4.4 AES procedure to perform a cipher operation

Introduction

A typical cipher operation is explained below. Detailed information is provided in sections starting from *Section 21.4.8: AES basic chaining modes (ECB, CBC)*.

Initialization of AES

To initialize AES, first disable it by clearing the EN bit of the AES_CR register. Then perform the following steps in any order:

- Configure the AES mode, by programming the MODE[1:0] bitfield of the AES_CR register.
  - For encryption, select Mode 1 (MODE[1:0] = 00).
  - For decryption, select Mode 3 (MODE[1:0] = 10), unless ECB or CBC chaining modes are used. In this latter case, perform an initial key derivation of the encryption key, as described in *Section 21.4.5: AES decryption round key preparation*.
- Select the chaining mode, by programming the CHMOD[2:0] bitfield of the AES_CR register.
- Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE[1:0] bitfield in the AES_CR register.
- When it is required (for example in CBC or CTR chaining modes), write the initialization vector into the AES_IVRx registers.
- Configure the key size (128-bit or 256-bit), with the KEYSIZE bitfield of the AES_CR register.
- Write a symmetric key into the AES_KEYRx registers (4 or 8 registers depending on the key size).

Data append

This section describes different ways of appending data for processing, where the size of data to process is not a multiple of 128 bits.

For ECB or CBC mode, refer to *Section 21.4.6: AES ciphertext stealing and data padding*. The last block management in these cases is more complex than in the sequence described in this section.

Data append through polling

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral by setting the EN bit of the AES_CR register.
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the AES_DINR register.
   b) Wait until the status flag CCF is set in the AES_SR, then read the four data words from the AES_DOUTR register.
   c) Clear the CCF flag, by setting the CCFC bit of the AES_CR register.
   d) If the data block just processed is the second-last block of the message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB bitfield of the AES_CR register, for AES to compute a correct tag;.
3. As it is the last block, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES_CR register.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the AES_DINR register, to allow sending the key to the AES processor.

NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.

Data append using interrupt

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES by setting the CCFIE bit of the AES_CR register.
2. Enable the AES peripheral by setting the EN bit of the AES_CR register.
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine, upon interrupt:
   a) Read four output data words from the AES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCFC bit of the AES_CR register.
   c) If the data block just processed is the second-last block of an message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB bitfield of the AES_CR register, for AES to compute a correct tag;. Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES_CR register and quit the interrupt service routine.
   e) Write next four input data words into the AES_DINR register and quit the interrupt service routine.

Note: AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations.

NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.

Data append using DMA

With this method, all the transfers and processing are managed by DMA and AES. To use the method, proceed as follows:

1. Prepare the last four-word data block (if the data to process does not fill it completely), by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in *Section 21.4.16: AES DMA interface*. Configure the DMA controller so as to generate an interrupt on transfer completion. In case of GCM payload encryption or CCM payload decryption, DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Data append through polling* must be used instead for this last block, because NPBLB bits must be setup before processing the block, for AES to compute a correct tag.
3. Enable the AES peripheral by setting the EN bit of the AES_CR register
4. Enable DMA requests by setting the DMAINEN and DMAOUTEN bits of the AES_CR register.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

Note: The CCF flag has no use with this method, because the reading of the AES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

NPBLB bits are not used in header phase of GCM, GMAC, and CCM chaining modes.

21.4.5 AES decryption round key preparation

Internal key schedule is used to generate AES round keys. In AES encryption, the round 0 key is the one stored in the key registers. AES decryption must start using the last round key. As the encryption key is stored in memory, a special key scheduling must be performed to obtain the decryption key. This key scheduling is only required for AES decryption in ECB and CBC modes.

Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES_CR (key process only), then proceed with the decryption by setting MODE[1:0] to 10 (Mode 3, decryption only). Mode 2 usage is described below:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES_CR. The CHMOD[2:0] bitfield is not significant in this case because this key derivation mode is independent of the chaining algorithm selected.
3. Set key length to 128 or 256 bits, via KEYSIZE bit of AES_CR register.
4. Write the AES_KEYRx registers (128 or 256 bits) with encryption key. Writes to the AES_IVRx registers have no effect.
5. Enable the AES peripheral, by setting the EN bit of the AES_CR register.
6. Wait until the CCF flag is set in the AES_SR register.
7. Clear the CCF flag. Derived key is available in AES core, ready to use for decryption.

Note: The AES is disabled by hardware when the derivation key is available.

To restart a derivation key computation, repeat steps 4, 5, 6, and 7.

Note: The operation of the key preparation lasts 59 or 82 clock cycles, depending on the key size (128- or 256-bit).

21.4.6 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (128 bits), ciphertext stealing techniques are used, such as those described in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since the AES peripheral does not support such techniques, the application must complete the last block of input data using data from the second last block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, when AES is used in other modes than ECB or CBC, an incomplete input data block (that is, block with input data shorter than 128 bits) must be padded with zeros prior to encryption (that is, extra bits must be appended to the trailing end of the data string). After decryption, the extra bits must be discarded. As AES does not implement automatic data padding operation to **the last block**, the application must follow the recommendation given in *Section 21.4.4: AES procedure to perform a cipher operation on page 551* to manage messages the size of which is not a multiple of 128 bits.

Note: Padding data are swapped in a similar way as normal data, according to the DATATYPE[1:0] field of the AES_CR register (see Section 21.4.13: AES data registers and data swapping for details).

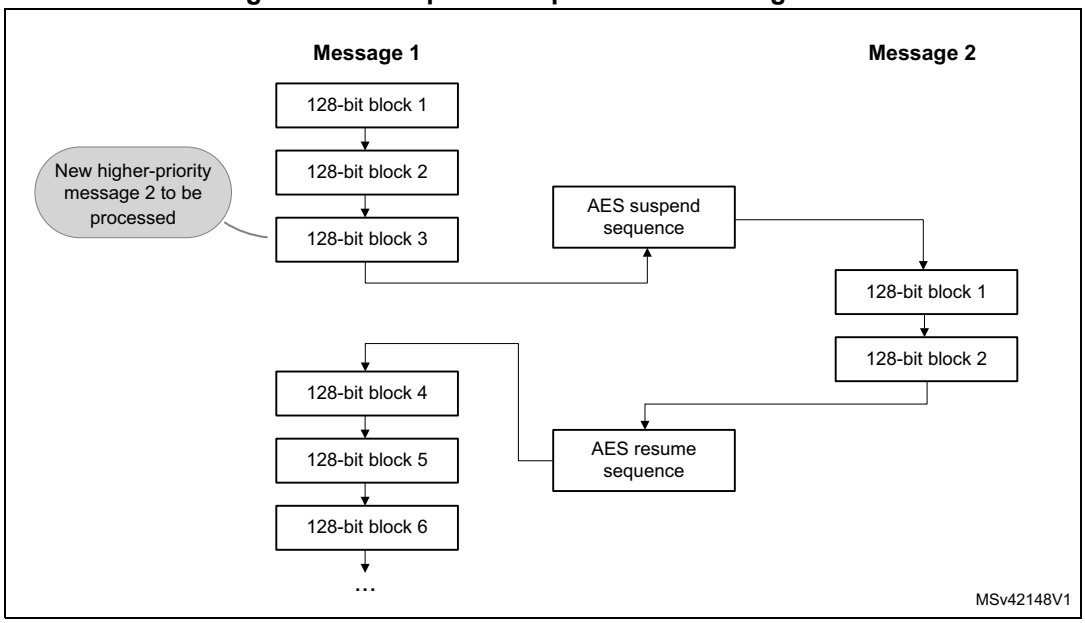
21.4.7 AES task suspend and resume

A message can be suspended if another message with a higher priority must be processed. When this highest priority message is sent, the suspended message can resume in both encryption or decryption mode.

Suspend/resume operations do not break the chaining operation and the message processing can resume as soon as AES is enabled again to receive the next data block.

Figure 94 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 94. Example of suspend mode management



A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

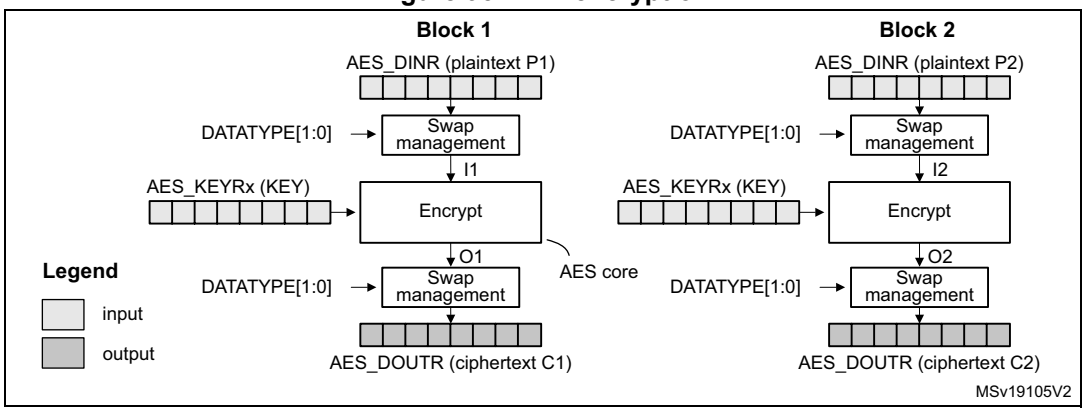
21.4.8 AES basic chaining modes (ECB, CBC)

Overview

This section gives a brief explanation of the four basic operation modes provided by the AES core: ECB encryption, ECB decryption, CBC encryption and CBC decryption. For detailed information, refer to the FIPS publication 197 from November 26, 2001.

Figure 95 illustrates the electronic codebook (ECB) encryption.

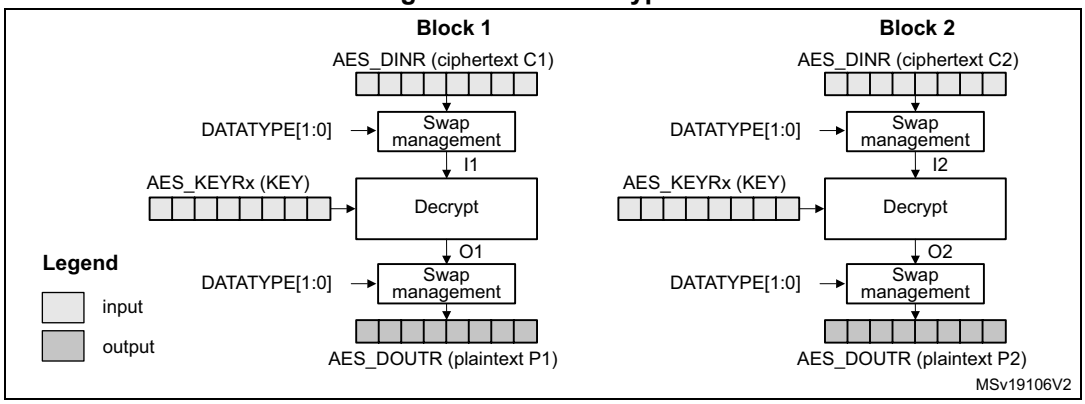
Figure 95. ECB encryption



In ECB encrypt mode, the 128-bit plaintext input data block Px in the AES_DINR register first goes through bit/byte/half-word swapping. The swap result Ix is processed with the AES core set in encrypt mode, using a 128- or 256-bit key. The encryption result Ox goes through bit/byte/half-word swapping, then is stored in the AES_DOUTR register as 128-bit ciphertext output data block Cx. The ECB encryption continues in this way until the last complete plaintext block is encrypted.

Figure 96 illustrates the electronic codebook (ECB) decryption.

Figure 96. ECB decryption

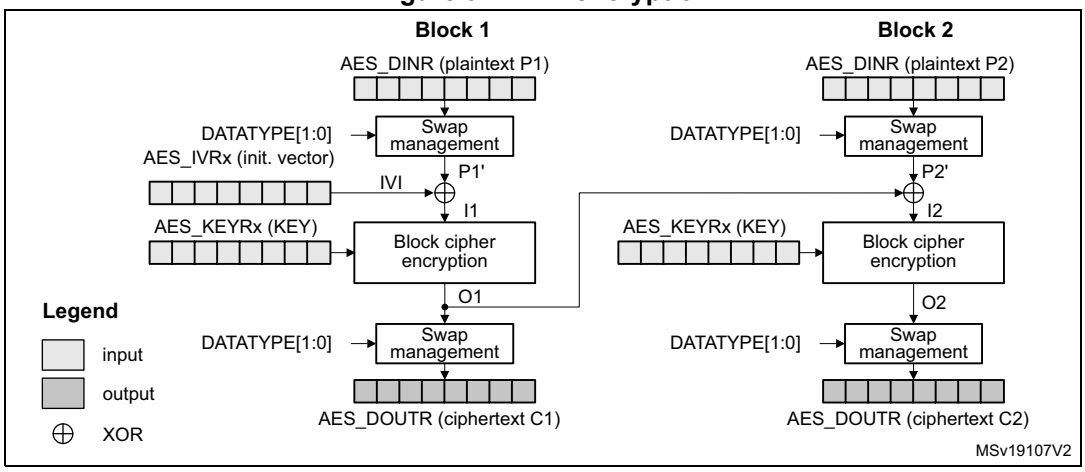


To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported by the AES core.

In ECB decrypt mode, the 128-bit ciphertext input data block C1 in the AES_DINR register first goes through bit/byte/half-word swapping. The keying sequence is reversed compared to that of the ECB encryption. The swap result I1 is processed with the AES core set in decrypt mode, using the formerly prepared decryption key. The decryption result goes through bit/byte/half-word swapping, then is stored in the AES_DOUTR register as 128-bit plaintext output data block P1. The ECB decryption continues in this way until the last complete ciphertext block is decrypted.

Figure 97 illustrates the cipher block chaining (CBC) encryption.

Figure 97. CBC encryption

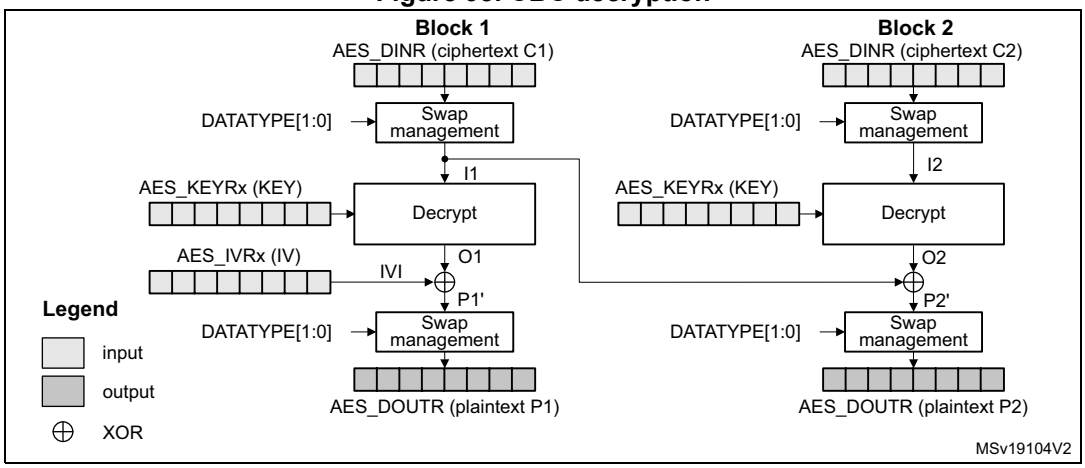


In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after swapping operation, is used as ciphertext C1. The O1 data is then XOR-ed with the second-block plaintext data P2' to produce the I2 input data for the AES core to produce the second block of ciphertext data. The chaining of data blocks continues in this way until the last plaintext block in the message is encrypted.

If the message size is not a multiple of 128 bits, the final partial data block is encrypted in the way explained in *Section 21.4.6: AES ciphertext stealing and data padding*.

Figure 98 illustrates the cipher block chaining (CBC) decryption.

Figure 98. CBC decryption



In CBC decrypt mode, like in ECB decrypt mode, the secret key must be prepared to perform an AES decryption.

After the key preparation process, the decryption goes as follows: the first 128-bit ciphertext block (after the swap operation) is used directly as the AES core input block I1 for decrypt operation, using the 128-bit or 256-bit key. Its output O1 is XOR-ed with the 128-bit IVI field (that must be identical to that used during encryption) to produce the first plaintext block P1.

The second ciphertext block is processed in the same way as the first block, except that the I1 data from the first block is used in place of the initialization vector.

The decryption continues in this way until the last complete ciphertext block is decrypted.

If the message size is not a multiple of 128 bits, the final partial data block is decrypted in the way explained in *Section 21.4.6: AES ciphertext stealing and data padding*.

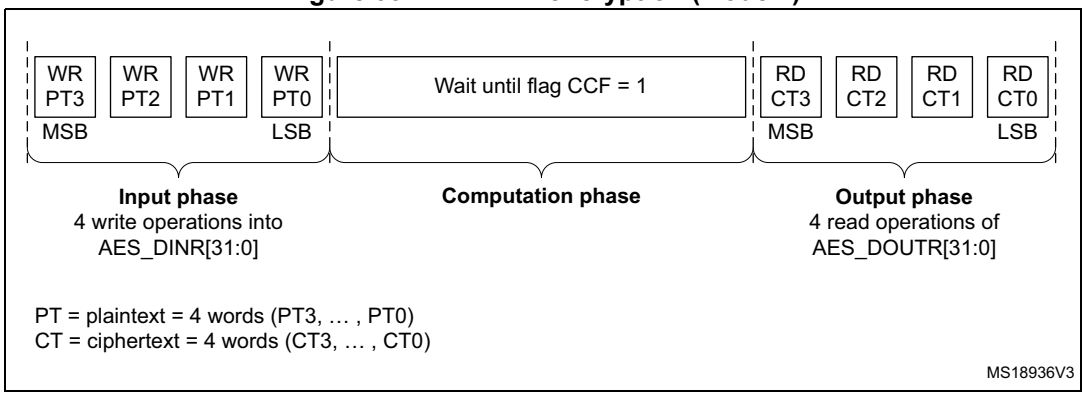
For more information on data swapping, refer to *Section 21.4.13: AES data registers and data swapping*.

ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in *Section 21.4.4*):

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select the Mode 1 by setting to 00 the MODE[1:0] bitfield of the AES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES_CR register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield.
3. Select 128- or 256-bit key length through the KEYSIZE bit of the AES_CR register.
4. Write the AES_KEYRx registers (128 or 256 bits) with encryption key. Fill the AES_IVRx registers with the initialization vector data if CBC mode has been selected.
5. Enable the AES peripheral by setting the EN bit of the AES_CR register.
6. Write the AES_DINR register four times to input the plaintext (MSB first), as shown in *Figure 99*.
7. Wait until the CCF flag is set in the AES_SR register.
8. Read the AES_DOUTR register four times to get the ciphertext (MSB first) as shown in *Figure 99*. Then clear the CCF flag by setting the CCFC bit of the AES_CR register.
9. Repeat steps *6-7-8* to process all the blocks with the same encryption key.

Figure 99. ECB/CBC encryption (Mode 1)



ECB/CBC decryption sequence

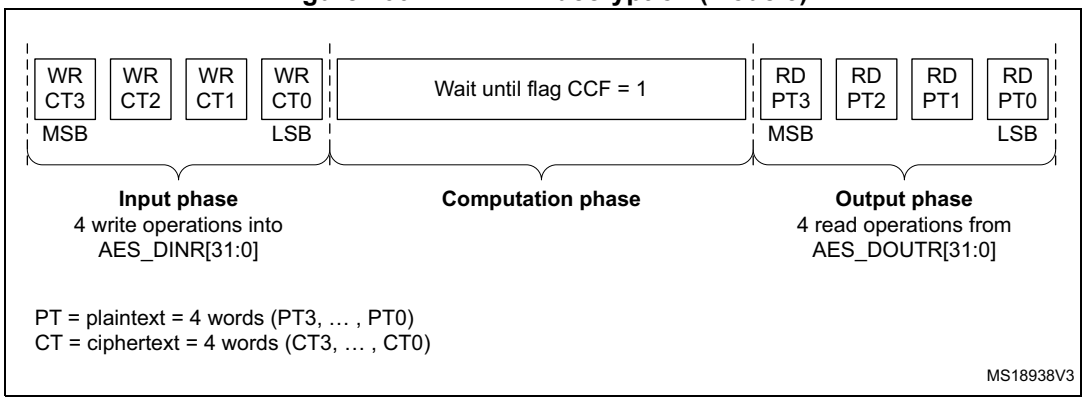
The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in *Section 21.4.4*).

1. Follow the steps described in *Section 21.4.5: AES decryption round key preparation*, in order to prepare the decryption key in AES core.
2. Select the Mode 3 by setting to 10 the MODE[1:0] bitfield of the AES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES_CR

register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield. KEYSIZE bitfield must be kept as-is.

3. Write the AES_IVRx registers with the initialization vector (required in CBC mode only).
4. Enable AES by setting the EN bit of the AES_CR register.
5. Write the AES_DINR register four times to input the cipher text (MSB first), as shown in *Figure 100*.
6. Wait until the CCF flag is set in the AES_SR register.
7. Read the AES_DOUTR register four times to get the plain text (MSB first), as shown in *Figure 100*. Then clear the CCF flag by setting the CCFC bit of the AES_CR register.
8. Repeat steps *5*-*6*-*7* to process all the blocks encrypted with the same key.

Figure 100. ECB/CBC decryption (Mode 3)



Suspend/resume operations in ECB/CBC modes

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register.
2. If DMA is not used, read four times the AES_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.
3. If DMA is not used, poll the CCF flag of the AES_SR register until it becomes 1 (computation completed).
4. Clear the CCF flag by setting the CCFC bit of the AES_CR register.
5. Save initialization vector registers (only required in CBC mode as AES_IVRx registers are altered during the data processing).
6. Disable the AES peripheral by clearing the bit EN of the AES_CR register.
7. Save the AES_CR register and clear the key registers if they are not needed, to process the higher priority message.
8. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, and so on).

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Restore AES_CR register (with correct KEYSIZE) then restore AES_KEYRx registers.
4. Prepare the decryption key as described in *Section 21.4.5: AES decryption round key preparation* (only required for ECB or CBC decryption).
5. Restore AES_IVRx registers using the saved configuration (only required in CBC mode).
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.
7. If DMA is used, enable AES DMA transfers by setting the DMAINEN and DMAOUTEN bits of the AES_CR register.

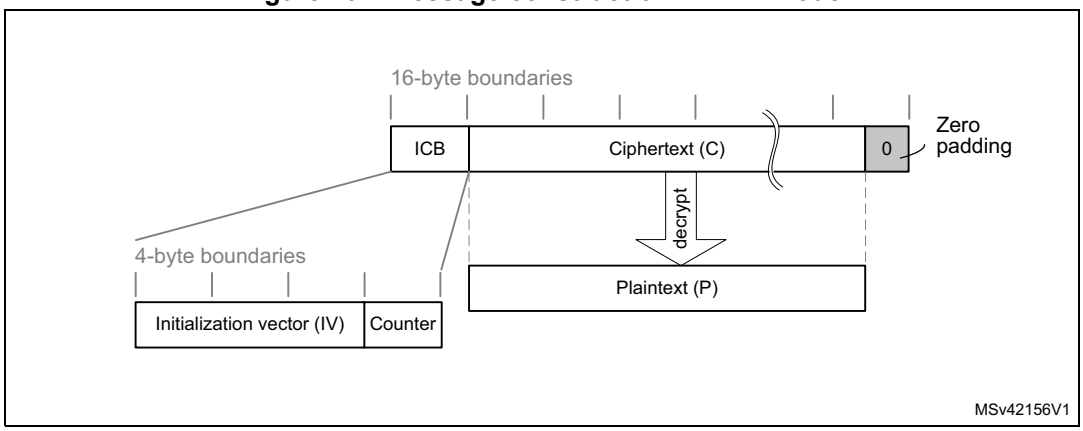
21.4.9 AES counter (CTR) mode

Overview

The counter mode (CTR) uses AES as a key-stream generator. The generated keys are then XOR-ed with the plaintext to obtain the ciphertext.

CTR chaining is defined in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.* A typical message construction in CTR mode is given in *Figure 101*.

Figure 101. Message construction in CTR mode



The structure of this message is:

- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

CTR encryption and decryption

Figure 102 and *Figure 103* describe the CTR encryption and decryption process, respectively, as implemented in the AES peripheral. The CTR mode is selected by writing 010 to the CHMOD[2:0] bitfield of AES_CR register.

Figure 102. CTR encryption

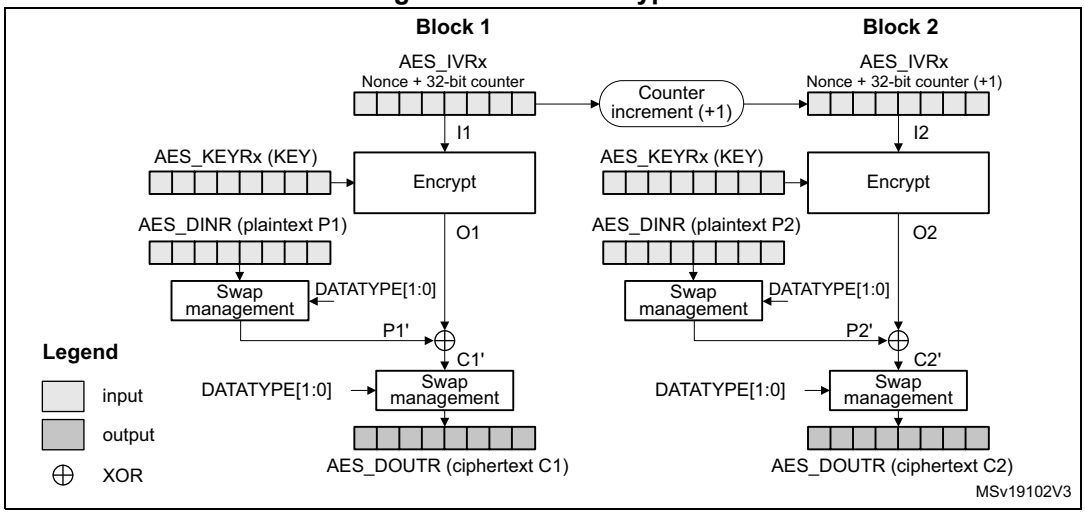
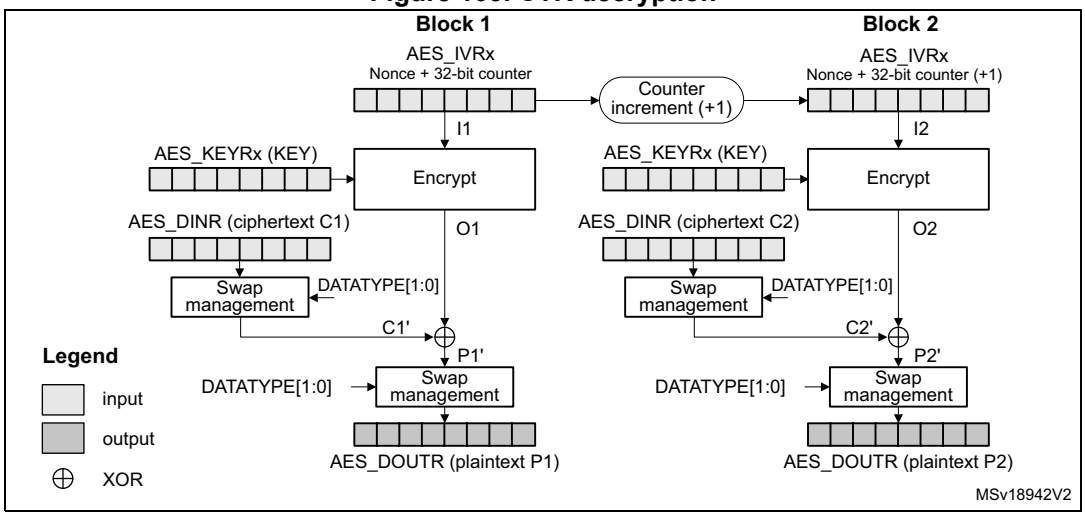


Figure 103. CTR decryption



In CTR mode, the cryptographic core output (also called keystream) Ox is XOR-ed with relevant input block (Px' for encryption, Cx' for decryption), to produce the correct output block (Cx' for encryption, Px' for decryption). Initialization vectors in AES must be initialized as shown in *Table 122*.

Table 122. CTR mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0} 32-bit counter = 0x0001

Unlike in CBC mode that uses the AES_IVRx registers only once when processing the first data block, in CTR mode AES_IVRx registers are used for processing each data block, and the AES peripheral increments the counter bits of the initialization vector (leaving the nonce bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that is then XOR-ed with the plaintext (CTR encryption) or ciphertext (CTR decryption) input. In CTR mode, the MODE[1:0] bitfield setting 01 (key derivation) is forbidden and all the other settings default to encryption mode.

The sequence of events to perform an encryption or a decryption in CTR chaining mode:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select CTR chaining mode by setting to 010 the CHMOD[2:0] bitfield of the AES_CR register. Set MODE[1:0] bitfield to any value other than 01.
3. Initialize the AES_KEYRx registers, and load the AES_IVRx registers as described in *Table 122*.
4. Set the EN bit of the AES_CR register, to start encrypting the current counter (EN is automatically reset when the calculation finishes).
5. If it is the last block, pad the data with zeros to have a complete block, if needed.
6. Append data in AES, and read the result. The three possible scenarios are described in *Section 21.4.4: AES procedure to perform a cipher operation*.
7. Repeat the previous step till the second-last block is processed. For the last block, apply the two previous steps and discard the bits that are not part of the payload (if the size of the significant data in the last input block is less than 16 bytes).

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, and resume the message that was interrupted. Detailed CBC suspend/resume sequence is described in *Section 21.4.8: AES basic chaining modes (ECB, CBC)*.

Note: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.

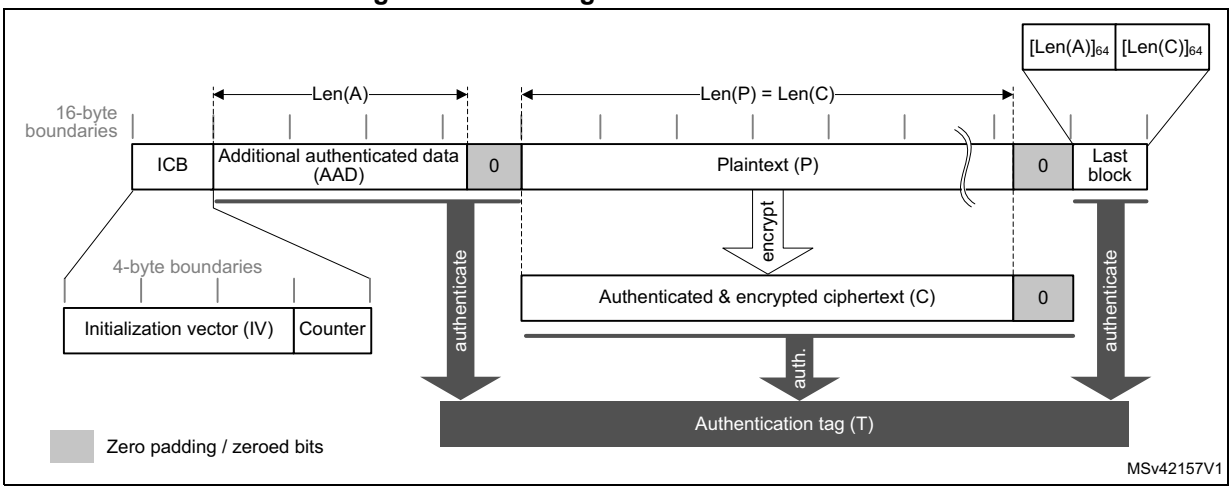
21.4.10 AES Galois/counter mode (GCM)

Overview

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, GCM algorithm is based on AES counter mode. It uses a multiplier over a fixed finite field to generate the tag.

GCM chaining is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.* A typical message construction in GCM mode is given in *Figure 104*.

Figure 104. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. Note that the GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed 2^{32} - 2 128-bit blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 123*.

The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

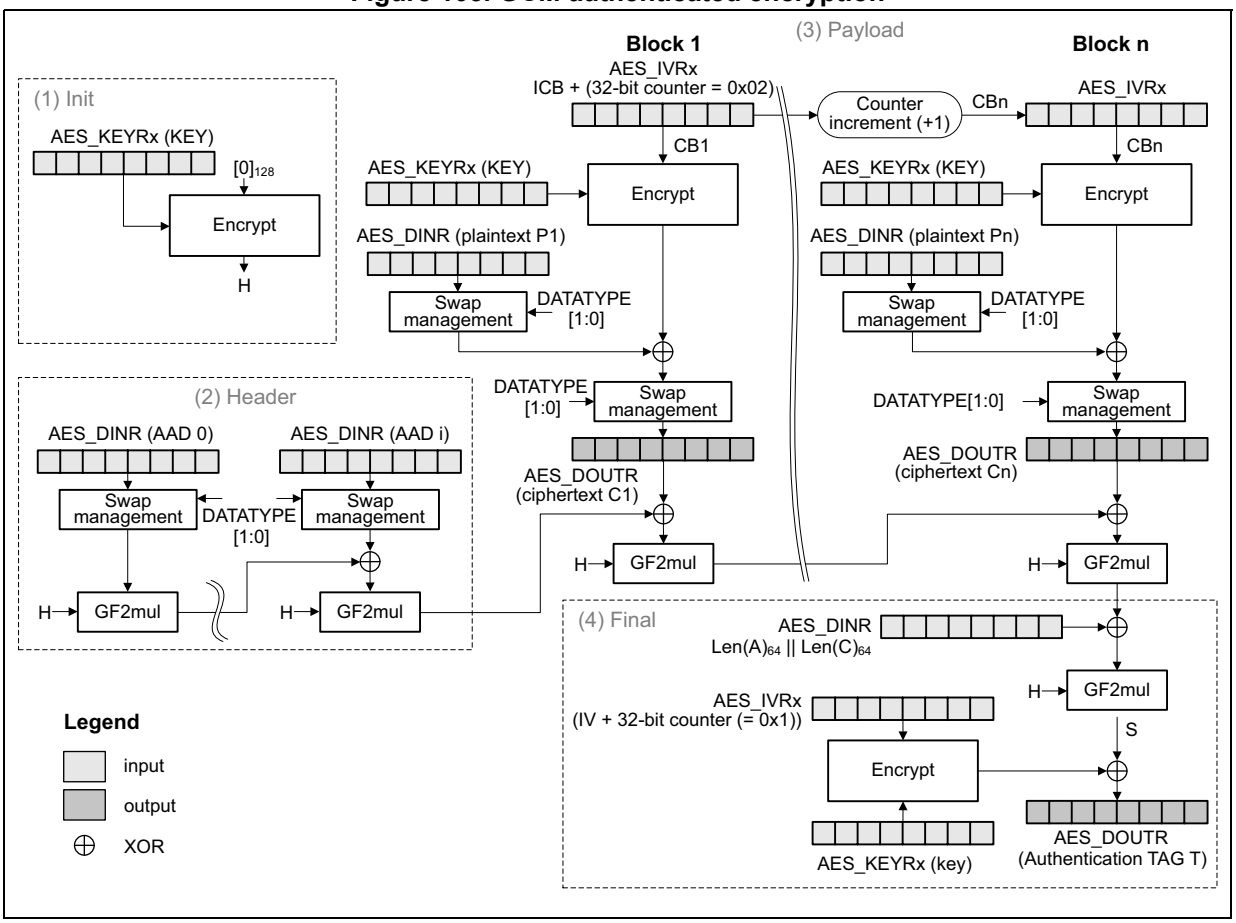
Table 123. GCM last block definition

Endianness	Bit[0] ---------- Bit[31]	Bit[32]---------- Bit[63]	Bit[64] -------- Bit[95]	Bit[96] --------- Bit[127]
Input data	0x0	AAD length[31:0]	0x0	Payload length[31:0]

GCM processing

Figure 105 describes the GCM implementation in the AES peripheral. The GCM is selected by writing 011 to the CHMOD[2:0] bitfield of the AES_CR register.

Figure 105. GCM authenticated encryption



The mechanism for the confidentiality of the plaintext in GCM mode is similar to that in the Counter mode, with a particular increment function (denoted 32-bit increment) that generates the sequence of input counter blocks.

AES_IVRx registers keeping the **counter block** of data are used for processing each data block. The AES peripheral automatically increments the Counter[31:0] bitfield. The first counter block (CB1) is derived from the initial counter block ICB by the application software (see *Table 124*).

Table 124. Initialization of AES_IVRx registers in GCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

Note: In this mode, the setting 01 of the MODE[1:0] bitfield (key derivation) is forbidden.

The authentication mechanism in GCM mode is based on a hash function called **GF2mul** that performs multiplication by a fixed parameter, called hash subkey (H), within a binary Galois field.

A GCM message is processed through the following phases, further described in next subsections:

- **Init phase**: AES prepares the GCM hash subkey (H).
- **Header phase**: AES processes the additional authenticated data (AAD), with hash computation only.
- **Payload phase**: AES processes the plaintext (P) with hash computation, counter block encryption and data XOR-ing. It operates in a similar way for ciphertext (C).
- **Final phase**: AES generates the authenticated tag (T) using the last block of the message.

GCM init phase

During this first step, the GCM hash subkey (H) is calculated and saved internally, to be used for processing all the blocks. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select GCM chaining mode, by setting to 011 the CHMOD[2:0] bitfield of the AES_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES_CR register.
4. Set the MODE[1:0] bitfield of the AES_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES_KEYRx registers with a key, and initialize AES_IVRx registers with the information as defined in *Table 124*.
6. Start the calculation of the hash key, by setting to 1 the EN bit of the AES_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag of the AES_SR register, by setting the CCFC bit of the AES_CR register.

GCM header phase

This phase coming after the GCM Init phase must be completed before the payload phase. The sequence to execute, identical for encryption and decryption, is:

1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. Enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways described in *Section 21.4.4: AES procedure to perform a cipher operation*. No data is read during this phase.
4. Repeat the step *3* until the last additional authenticated data block is processed.

Note: The header phase can be skipped if there is no AAD, that is, Len(A) = 0.

GCM payload phase

This phase, identical for encryption and decryption, is executed after the GCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES_DOUTR register. The sequence to execute is:

1. Indicate the payload phase, by setting to 10 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in *Section 21.4.4: AES procedure to perform a cipher operation on page 551*, and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block, discard the bits that are not part of the payload when the last block size is less than 16 bytes.

Note: The payload phase can be skipped if there is no payload data, that is, Len(C) = 0 (see GMAC mode).

GCM final phase

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCMPH[1:0] bitfield of the AES_CR register.
2. Compose the data of the block, by concatenating the AAD bit length and the payload bit length, as shown in *Table 123*. Write the block into the AES_DINR register.
3. Wait until the end of computation, indicated by the CCF flag of the AES_SR transiting to 1.
4. Get the GCM authentication tag, by reading the AES_DOUTR register four times.
5. Clear the CCF flag of the AES_SR register, by setting the CCFC bit of the AES_CR register.
6. Disable the AES peripheral, by clearing the bit EN of the AES_CR register. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data is written to AES_DINR normally (no swapping), while swapping is applied to tag data read from AES_DOUTR.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES_SR register set to 1.
2. In the payload phase, if DMA is not used, read four times the AES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.
3. Clear the CCF flag of the AES_SR register, by setting the CCFC bit of the AES_CR register.
4. Save the AES_SUSPxR registers in the memory, where x is from 0 to 7.
5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.
7. Save the current AES configuration in the memory, excluding the initialization vector registers AES_IVRx. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPxR registers, where x is from 0 to 7.
4. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
5. Restore the initial setting values in the AES_CR and AES_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.

If DMA is used, enable AES DMA requests by setting the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES_CR register.

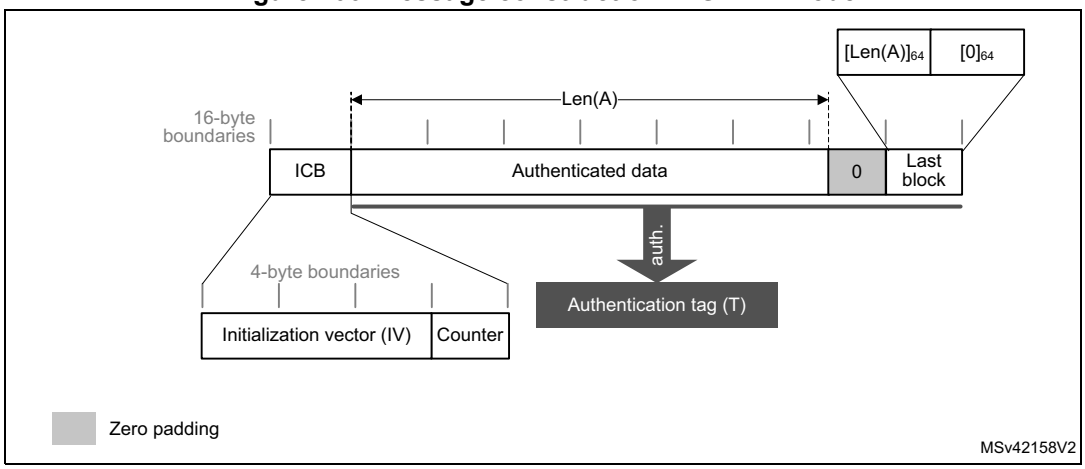
21.4.11 AES Galois message authentication code (GMAC)

Overview

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code). It is based on GCM algorithm, as defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

A typical message construction for GMAC is given in *Figure 106*.

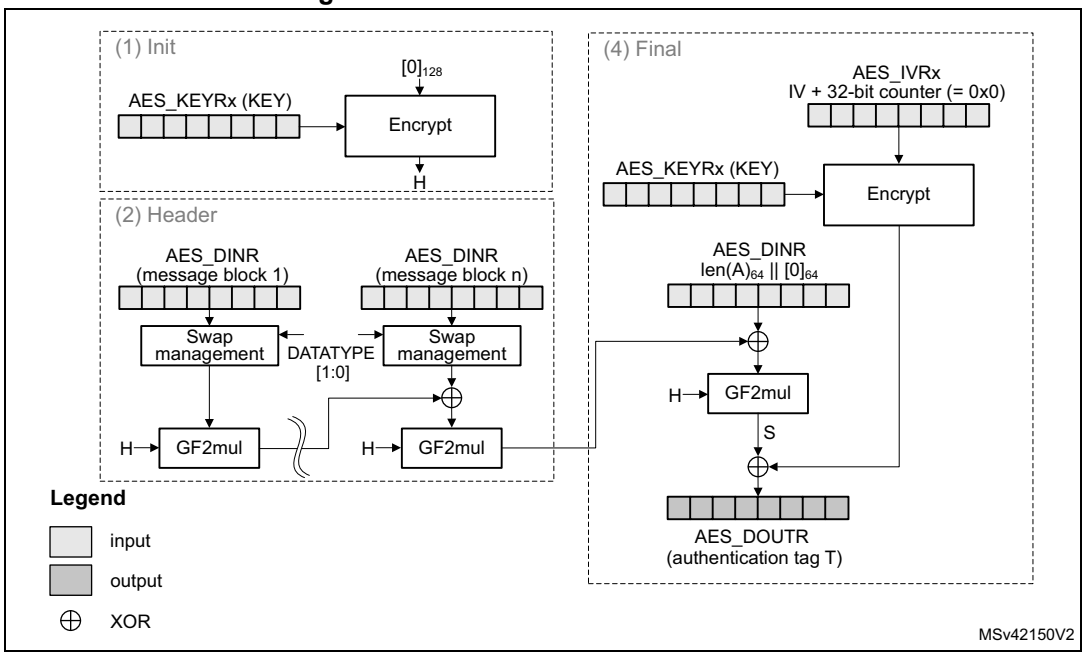
Figure 106. Message construction in GMAC mode



AES GMAC processing

Figure 107 describes the GMAC mode implementation in the AES peripheral. This mode is selected by writing 011 to the CHMOD[2:0] bitfield of the AES_CR register.

Figure 107. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message only containing a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

21.4.12 AES counter with CBC-MAC (CCM)

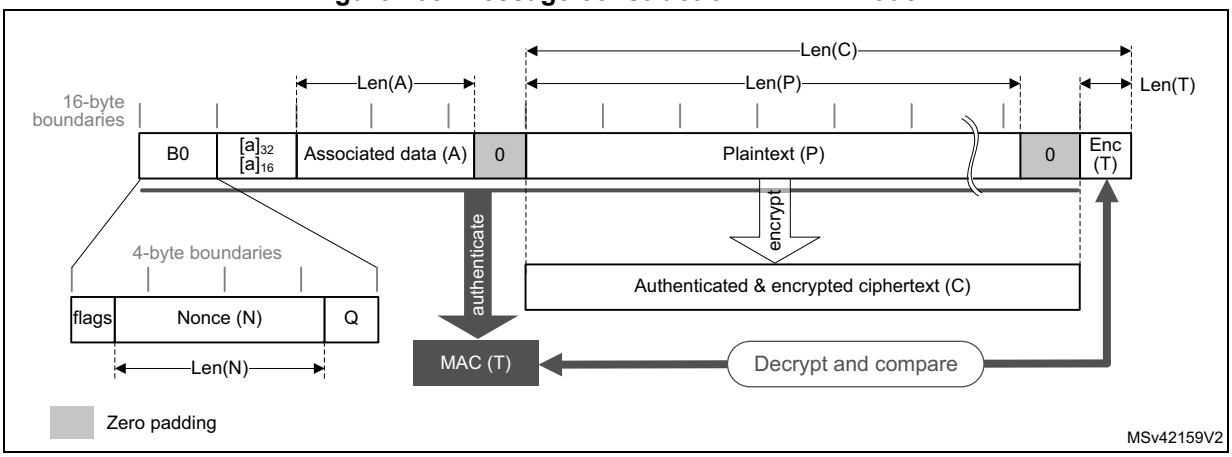
Overview

The AES **counter with cipher block chaining-message authentication code** (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES in counter mode. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve this CBC-MAC as an authentication mode outside the context of the CCM specification.

CCM chaining is specified in NIST *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*. A typical message construction for CCM is given in *Figure 108*.

Figure 108. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the Associated Data (A).
  This part of the message is only authenticated, not encrypted. This section has a

known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 108*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:

– If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
– If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
– If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 108*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

CTR0: 0710111213 141516 00000000 00000000

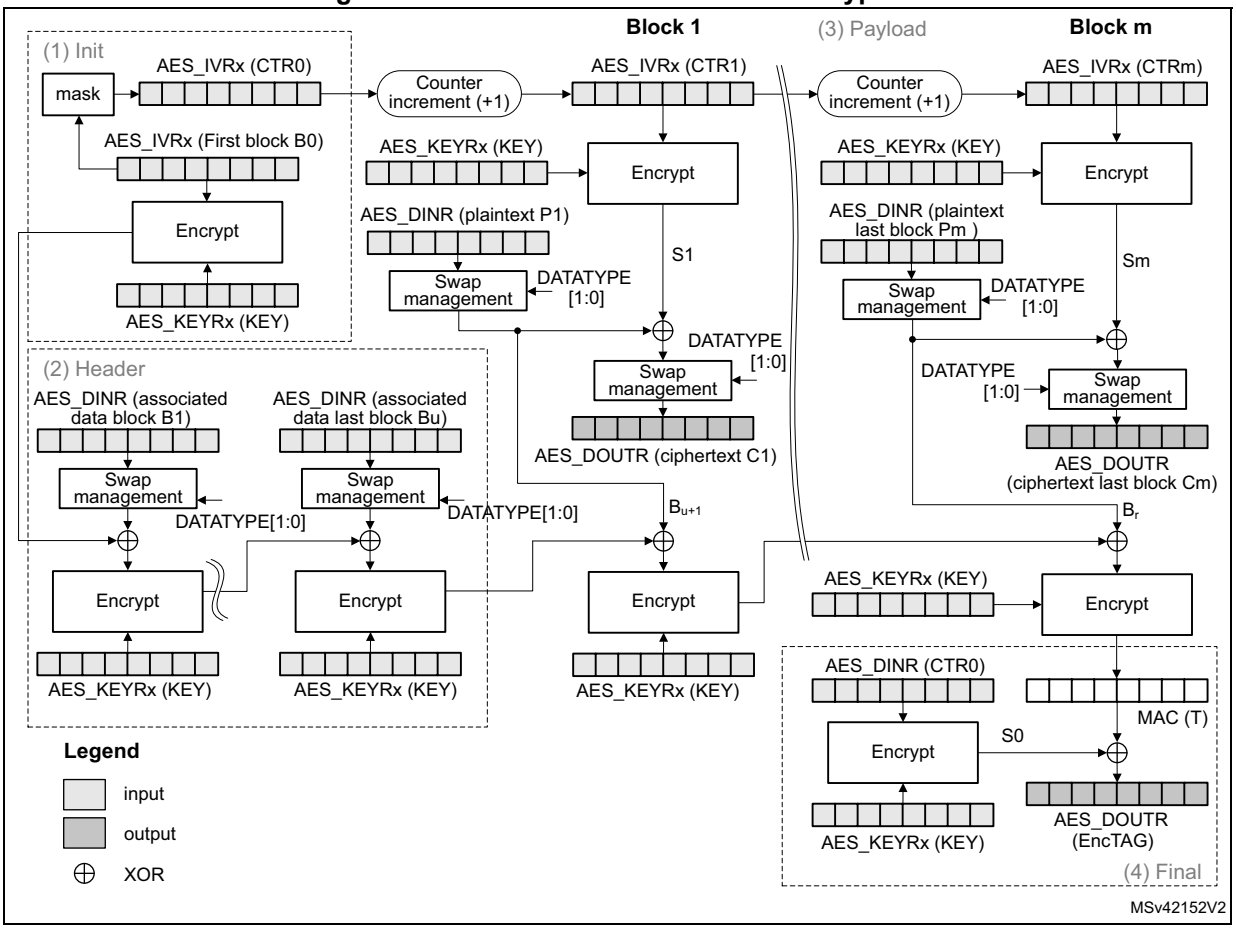
CTR1: 0710111213 141516 00000000 00000001

Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.

CCM processing

Figure 109 describes the CCM implementation within the AES peripheral (encryption example). This mode is selected by writing 100 into the CHMOD[2:0] bitfield of the AES_CR register.

Figure 109. CCM mode authenticated encryption



The data input to the generation-encryption process are a valid nonce, a valid payload string, and a valid associated data string, all properly formatted. The CBC chaining mechanism is applied to the formatted plaintext data to generate a MAC, with a known length. Counter mode encryption that requires a sufficiently long sequence of counter blocks as input, is applied to the payload string and separately to the MAC. The resulting ciphertext C is the output of the generation-encryption process on plaintext P.

AES_IVRx registers are used for processing each data block, AES automatically incrementing the CTR counter with a bit length defined by the first block B0. *Table 125* shows how the application must load the B0 data.

Note: The AES peripheral in CCM mode supports counters up to 64 bits, as specified by NIST.

Table 125. Initialization of AES_IVRx registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96]	B0[95:64]	B0[63:32]	B0[31:0]

Note: In this mode, the setting 01 of the MODE[1:0] bitfield (key derivation) is forbidden.

A CCM message is processed through the following phases, further described in next subsections:

- **Init phase**: AES processes the first block and prepares the first counter block.
- **Header phase**: AES processes associated data (A), with tag computation only.
- **Payload phase**: IP processes plaintext (P), with tag computation, counter block encryption, and data XOR-ing. It works in a similar way for ciphertext (C).
- **Final phase**: AES generates the message authentication code (MAC).

CCM Init phase

In this phase, the first block B0 of the CCM message is written into the AES_IVRx register. The AES_DOUTR register does not contain any output data. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select CCM chaining mode, by setting to 100 the CHMOD[2:0] bitfield of the AES_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES_CR register.
4. Set the MODE[1:0] bitfield of the AES_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES_KEYRx registers with a key, and initialize AES_IVRx registers with B0 data as described in *Table 125*.
6. Start the calculation of the counter, by setting to 1 the EN bit of the AES_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag in the AES_SR register, by setting to 1 the CCFC bit of the AES_CR register.

CCM header phase

This phase coming after the GCM Init phase must be completed before the payload phase. During this phase, the AES_DOUTR register does not contain any output data.

The sequence to execute, identical for encryption and decryption, is:

1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. Enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways described in *Section 21.4.4: AES procedure to perform a cipher operation*. No data is read during this phase.
4. Repeat the step *3* until the last additional authenticated data block is processed.

Note: The header phase can be skipped if there is no associated data, that is, Len(A) = 0.

The first block of the associated data (B1) must be formatted by software, with the associated data length.

CCM payload phase (encryption or decryption)

This phase, identical for encryption and decryption, is executed after the CCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES_DOUTR register. The sequence to execute is:

1. Indicate the payload phase, by setting to 10 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last data block to encrypt and the plaintext size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in *Section 21.4.4: AES procedure to perform a cipher operation on page 551*, and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), apply the two previous steps. For the last block, discard the data that is not part of the payload when the last block size is less than 16 bytes.

Note: The payload phase can be skipped if there is no payload data, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM final phase

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCMPH[1:0] bitfield of the AES_CR register.
2. Wait until the end-of-computation flag CCF of the AES_SR register is set.
3. Read four times the AES_DOUTR register: the output corresponds to the CCM authentication tag.
4. Clear the CCF flag of the AES_SR register by setting the CCFC bit of the AES_CR register.
5. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.
6. For authenticated decryption, compare the generated encrypted tag with the encrypted tag padded in the ciphertext.

Note: In this final phase, swapping is applied to tag data read from AES_DOUTR register.

When transiting from the header phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Application must mask the authentication tag output with tag length to obtain a valid tag.

Suspend/resume operations in CCM mode

To suspend the processing of a message in header or payload phase, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES_SR register set to 1.
2. In the payload phase, if DMA is not used, read four times the AES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the

AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.

3. Clear the CCF flag of the AES_SR register, by setting to 1 the CCFC bit of the AES_CR register.
4. Save the AES_SUSPxR registers (where x is from 0 to 7) in the memory.
5. Save the AES_IVRx registers as, during the data processing, they changed from their initial values.
6. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.
7. Save the current AES configuration in the memory, excluding the initialization vector registers AES_IVRx. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPxR registers (where x is from 0 to 7).
4. Write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers.
5. Restore the initial setting values in the AES_CR and AES_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.
7. If DMA is used, enable AES DMA requests by setting to 1 the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES_CR register.

21.4.13 AES data registers and data swapping

Data input and output

A 128-bit data block is entered into the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 128-bit data block is retrieved from the AES peripheral with four successive 32-bit word reads from the AES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The 32-bit data word for AES_DINR register or from AES_DOUTR register is organized in big endian order, that is:

- the most significant byte of a word to write into AES_DINR must be put on the lowest address out of the four adjacent memory locations keeping the word to write, or
- the most significant byte of a word read from AES_DOUTR goes to the lowest address out of the four adjacent memory locations receiving the word

For using DMA for input data block write into AES, the four words of the input block must be stored in the memory consecutively and in big-endian order, that is, the most significant word on the lowest address. See *Section 21.4.16: AES DMA interface*.

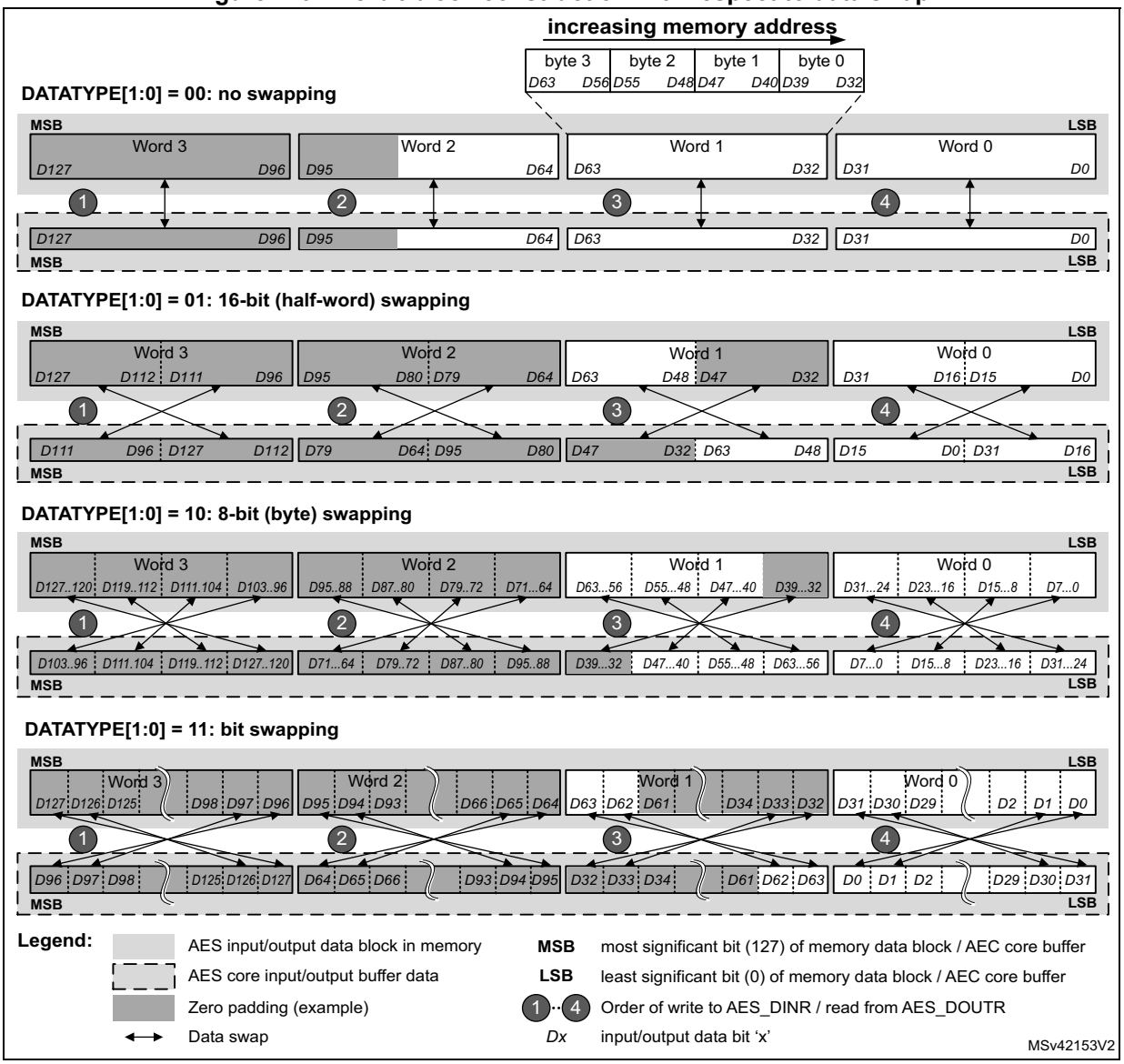
Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0] bitfield of the AES_CR register. The selection applies both to the input and the output of the AES core.

For different data swap types, *Figure 110* shows the construction of AES processing core input buffer data P127 to P0, from the input data entered through the AES_DINR register, or the construction of the output data available through the AES_DOUTR register, from the AES processing core output buffer data P127 to P0.

Figure 110. 128-bit block construction with respect to data swap



Note: The data in AES key registers (AES_KEYRx) and initialization registers (AES_IVRx) are not sensitive to the swap mode selection.

Data padding

Figure 110 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 48 message bits, with DATATYPE[1:0] = 01
- 56 message bits, with DATATYPE[1:0] = 10
- 34 message bits, with DATATYPE[1:0] = 11

21.4.14 AES key registers

The AES_KEYRx write-only registers store the encryption or decryption key bitfield KEY[127:0] or KEY[255:0]. The data to write to each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address (reads are not allowed for security reason).

The key is spread over eight registers as shown in *Table 126*.

Table 126. Key endianness in AES_KEYRx registers (128- or 256-bit key length)

AES_KEYR7 [31:0]	AES_KEYR6 [31:0]	AES_KEYR5 [31:0]	AES_KEYR4 [31:0]	AES_KEYR3 [31:0]	AES_KEYR2 [31:0]	AES_KEYR1 [31:0]	AES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key for encryption or decryption may be written into these registers when the AES peripheral is disabled, by clearing the EN bit of the AES_CR register.

The key registers are not affected by the data swapping controlled by DATATYPE[1:0] bitfield of the AES_CR register.

21.4.15 AES initialization vector registers

The four AES_IVRx registers keep the initialization vector input bitfield IVI[127:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address. The registers are also ordered from lowest address (AES_IVR0) to highest address (AES_IVR3).

The signification of data in the bitfield depends on the chaining mode selected. When used, the bitfield is updated upon each computation cycle of the AES core.

Write operations to the AES_IVRx registers when the AES peripheral is enabled have no effect to the register contents. For modifying the contents of the AES_IVRx registers, the EN bit of the AES_CR register must first be cleared.

Reading the AES_IVRx registers returns the latest counter value (useful for managing suspend mode).

The AES_IVRx registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield of the AES_CR register.

21.4.16 AES DMA interface

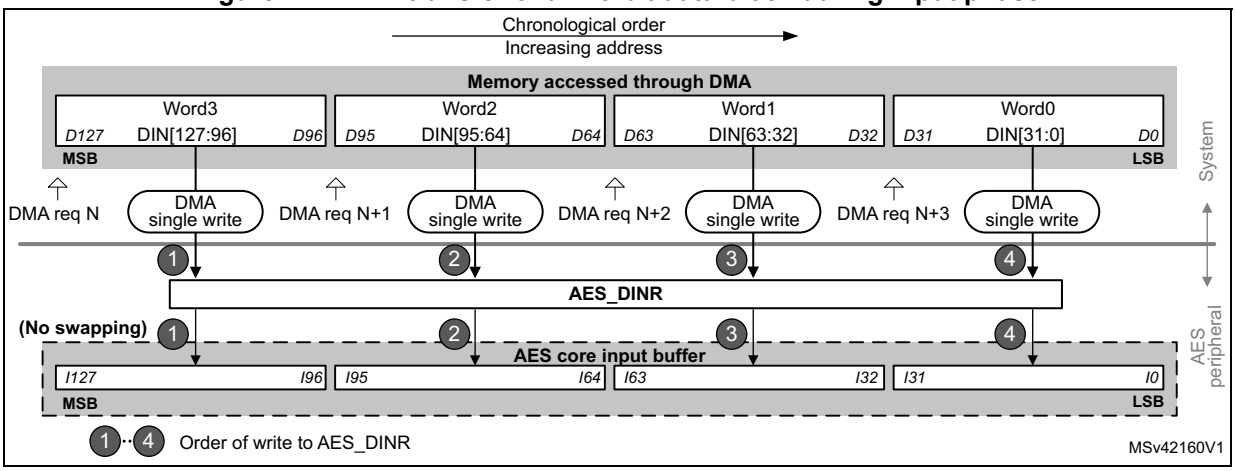
The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the AES_CR register.

Data input using DMA

Setting the DMAINEN bit of the AES_CR register enables DMA writing into AES. The AES peripheral then initiates a DMA request during the input phase each time it requires to write a 128-bit block (quadruple word) to the AES_DINR register, as shown in *Figure 111*.

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required. For example, in case of AES GCM encryption or AES CCM decryption, a DMA transfer must not include the last block. For details, refer to Section 21.4.4: AES procedure to perform a cipher operation.

Figure 111. DMA transfer of a 128-bit data block during input phase

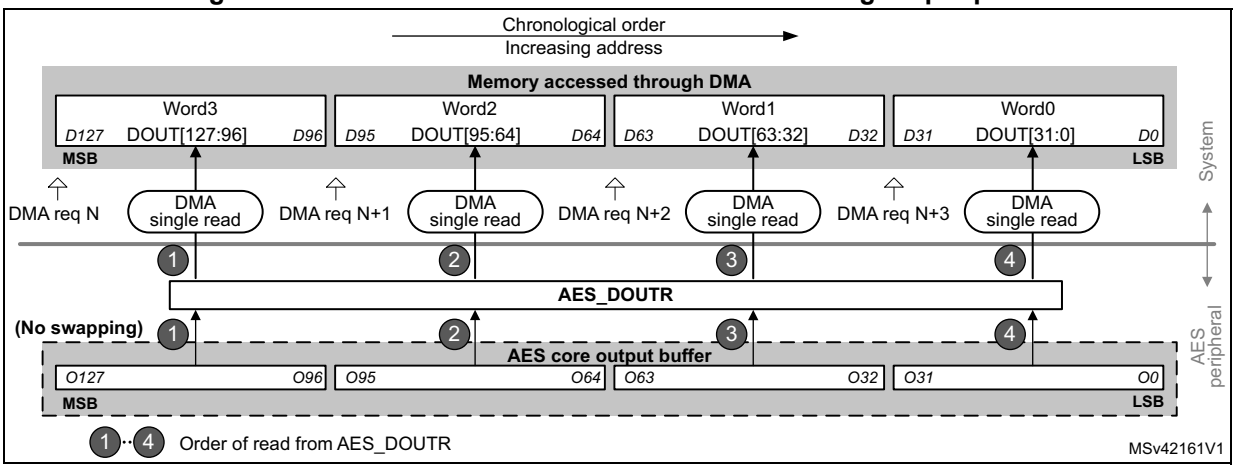


Data output using DMA

Setting the DMAOUTEN bit of the AES_CR register enables DMA reading from AES. The AES peripheral then initiates a DMA request during the Output phase each time it requires to read a 128-bit block (quadruple word) to the AES_DINR register, as shown in *Figure 112*.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Figure 112. DMA transfer of a 128-bit data block during output phase



DMA operation in different operating modes

DMA operations are usable when Mode 1 (encryption) or Mode 3 (decryption) are selected via the MODE[1:0] bitfield of the register AES_CR. As in Mode 2 (key derivation) the AES_KEYRx registers must be written by software, enabling the DMA transfer through the DMAINEN and DMAOUTEN bits of the AES_CR register have no effect in that mode.

DMA single requests are generated by AES until it is disabled. So, after the data output phase at the end of processing of a 128-bit data block, AES switches automatically to a new data input phase for the next data block, if any.

When the data transferring between AES and memory is managed by DMA, the CCF flag has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to data transferring managed by software. See *Section 21.4.4: AES procedure to perform a cipher operation*, subsection *Data append*, for details.

21.4.17 AES error management

AES configuration can be changed at any moment by clearing the EN bit of the AES_CR register.

Read error flag (RDERR)

Unexpected read attempt of the AES_DOUTR register sets the RDERR flag of the AES_SR register, and returns zero.

RDERR is triggered during the computation phase or during the input phase.

Note: AES is not disabled upon a RDERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES_CR register is set. For more details, refer to *Section 21.5: AES interrupts*.

The RDERR flag is cleared by setting the ERRIE bit of the AES_CR register.

Write error flag (WDERR)

Unexpected write attempt of the AES_DINR register sets the WRERR flag of the AES_SR register, and has no effect on the AES_DINR register. The WRERR is triggered during the computation phase or during the output phase.

Note: AES is not disabled after a WRERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES_CR register is set. For more details, refer to *Section 21.5: AES interrupts*.

The WRERR flag is cleared by setting the ERRC bit of the AES_CR register.

21.5 AES interrupts

Individual maskable interrupt sources generated by the AES peripheral signal the following events:

- computation completed
- read error
- write error

The individual sources are combined into the common interrupt signal aes_it that connects to NVIC (nested vectored interrupt controller). Each can individually be enabled/disabled, by setting/clearing the corresponding enable bit of the AES_CR register, and cleared by setting the corresponding bit of the AES_CR register.

The status of each can be read from the AES_SR register.

Table 127 gives a summary of the interrupt sources, their event flags and enable bits.

Table 127. AES interrupt requests

Interrupt acronym	AES interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCFC(1)
	read error flag	RDERR	ERRIE	set ERRC(1)
	write error flag	WRERR		

1. Bit of the AES_CR register.

21.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

Table 128. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Algorithm	Clock cycles
128-bit	Mode 1: Encryption	ECB, CBC, CTR	51
	Mode 2: Key derivation	-	59
	Mode 3: Decryption	ECB, CBC, CTR	51
256-bit	Mode 1: Encryption	ECB, CBC, CTR	75
	Mode 2: Key derivation	-	82
	Mode 3: Decryption	ECB, CBC, CTR	75

Table 129. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Algorithm	Init Phase	Header phase(1)	Payload phase(1)	Tag phase(1)
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	88	35	75	75
		CCM	87	79	162	82

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

21.7 AES registers

21.7.1 AES control register (AES_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
								rw	rw	rw	rw		rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

The bitfield sets the number of padding bytes in last block of payload:
0000: All bytes are valid (no padding)
0001: Padding for one least-significant byte of last block
...
1111: Padding for 15 least-significant bytes of last block

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the length of the key used in the AES cryptographic core, in bits:
0: 128
1: 256
Attempts to write the bit are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]:** GCM or CCM phase selection
This bitfield selects the phase of GCM, GMAC or CCM algorithm:
00: Init phase
01: Header phase
10: Payload phase
11: Final phase
The bitfield has no effect if other than GCM, GMAC or CCM algorithms are selected (through the ALGOMODE bitfield).

Bit 12 **DMAOUTEN**: DMA output enable
This bit enables/disables data transferring with DMA, in the output phase:
0: Disable
1: Enable
When the bit is set, DMA requests are automatically generated by AES during the output data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).

Bit 11 **DMAINEN**: DMA input enable
This bit enables/disables data transferring with DMA, in the input phase:
0: Disable
1: Enable
When the bit is set, DMA requests are automatically generated by AES during the input data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).

Bit 10 **ERRIE**: Error interrupt enable
This bit enables or disables (masks) the AES interrupt generation when RDERR and/or WRERR is set:
0: Disable (mask)
1: Enable

Bit 9 **CCFIE**: CCF interrupt enable
This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set:
0: Disable (mask)
1: Enable

Bit 8 **ERRC**: Error flag clear
Upon written to 1, this bit clears the RDERR and WRERR error flags in the AES_SR register:
0: No effect
1: Clear RDERR and WRERR flags
Reading the flag always returns zero.

Bit 7 **CCFC:** Computation complete flag clear
Upon written to 1, this bit clears the computation complete flag (CCF) in the AES_SR register:
0: No effect
1: Clear CCF
Reading the flag always returns zero.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode selection
This bitfield selects the AES chaining mode:
000: Electronic codebook (ECB)
001: Cipher-block chaining (CBC)
010: Counter mode (CTR)
011: Galois counter mode (GCM) and Galois message authentication code (GMAC)
100: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: AES operating mode
This bitfield selects the AES operating mode:
00: Mode 1: encryption
01: Mode 2: key derivation (or key preparation for ECB/CBC decryption)
10: Mode 3: decryption
11: Reserved
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type selection
This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register, through selecting the mode of data swapping:
00: None
01: Half-word (16-bit)
10: Byte (8-bit)
11: Bit
For more details, refer to *Section 21.4.13: AES data registers and data swapping*.
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: AES enable
This bit enables/disables the AES peripheral:
0: Disable
1: Enable
At any moment, clearing then setting the bit re-initializes the AES peripheral.
This bit is automatically cleared by hardware upon the completion of the key preparation (Mode 2) and upon the completion of GCM/GMAC/CCM initial phase.

21.7.2 AES status register (AES_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	WRERR	RDERR	CCF
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether AES is idle or busy during GCM payload **encryption** phase:

0: Idle

1: Busy

When the flag indicates “idle”, the current GCM encryption processing may be suspended to process a higher-priority message. In other chaining modes, or in GCM phases other than payload encryption, the flag must be ignored for the suspend process.

Bit 2 **WRERR**: Write error

This flag indicates the detection of an unexpected write operation to the AES_DINR register (during computation or data output phase):

0: Not detected

1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES_CR register.

The flag setting has no impact on the AES operation. Unexpected write is ignored.

Bit 1 **RDERR**: Read error flag

This flag indicates the detection of an unexpected read operation from the AES_DOUTR register (during computation or data input phase):

0: Not detected

1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES_CR register.

The flag setting has no impact on the AES operation. Unexpected read returns zero.

Bit 0 **CCF**: Computation completed flag

This flag indicates whether the computation is completed:

0: Not completed

1: Completed

The flag is set by hardware upon the completion of the computation. It is cleared by software, upon setting the CCFC bit of the AES_CR register.

Upon the flag setting, an interrupt is generated if enabled through the CCFIE bit of the AES_CR register.

The flag is significant only when the DMAOUTEN bit is 0. It may stay high when DMA_EN is 1.

21.7.3 AES data input register (AES_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

Only 32-bit access type is supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DIN[31:0]**: Input data word

A four-fold sequential write to this bitfield during the input phase results in writing a complete 128-bit block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 128-bit input buffer.

The data signification of the input data block depends on the AES operating mode:

- **Mode 1** (encryption): plaintext
- **Mode 2** (key derivation): the bitfield is not used (AES_KEYRx registers used for input)
- **Mode 3** (decryption): ciphertext

The data swap operation is described in *Section 21.4.13: AES data registers and data swapping on page 574*.

21.7.4 AES data output register (AES_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

Only 32-bit read access type is supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]**: Output data word

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF set), virtually reads a complete 128-bit block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

The data signification of the output data block depends on the AES operating mode:

- **Mode 1** (encryption): ciphertext
- **Mode 2** (key derivation): the bitfield is not used
- **Mode 3** (decryption): plaintext

The data swap operation is described in *Section 21.4.13: AES data registers and data swapping on page 574*.

21.7.5 AES key register 0 (AES_KEYR0)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

This write-only bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode:

- In **Mode 1** (encryption), **Mode 2** (key derivation): the value to write into the bitfield is the encryption key.
- In **Mode 3** (decryption): the value to write into the bitfield is the encryption key to be derived before being used for decryption.

The AES_KEYRx registers may be written only when KEYSIZE value is correct and when the AES peripheral is disabled (EN bit of the AES_CR register cleared). Note that, if, the key is directly loaded to AES_KEYRx registers (hence writes to key register is ignored and KEIF is set).

Refer to *Section 21.4.14: AES key registers on page 576* for more details.

21.7.6 AES key register 1 (AES_KEYR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

21.7.7 AES key register 2 (AES_KEYR2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

21.7.8 AES key register 3 (AES_KEYR3)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

21.7.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]
Refer to *Section 21.4.15: AES initialization vector registers on page 576* for description of the IVI[127:0] bitfield.
The initialization vector is only used in chaining modes other than ECB.
The AES_IVRx registers may be written only when the AES peripheral is disabled

21.7.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

21.7.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

21.7.12 AES initialization vector register 3 (AES_IVR3)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

21.7.13 AES key register 4 (AES_KEYR4)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

21.7.14 AES key register 5 (AES_KEYR5)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

21.7.15 AES key register 6 (AES_KEYR6)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

21.7.16 AES key register 7 (AES_KEYR7)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

Note: The key registers from 4 to 7 are used only when the key length of 256 bits is selected. They have no effect when the key length of 128 bits is selected (only key registers 0 to 3 are used in that case).

21.7.17 AES suspend registers (AES_SUSPxR)

Address offset: 0x040 + x * 0x4, (x = 0 to 7)

Reset value: 0x0000 0000

These registers contain the complete internal register states of the AES processor when the AES processing of the current task is suspended to process a higher-priority task.

Upon suspend, the software reads and saves the AES_SUSPxR register contents (where x is from 0 to 7) into memory, before using the AES processor for the higher-priority task.

Upon completion, the software restores the saved contents back into the corresponding suspend registers, before resuming the original task.

Note: These registers are used only when GCM, GMAC, or CCM chaining mode is selected.

These registers can be read only when AES is enabled. Reading these registers while AES is disabled returns 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUSP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SUSP[31:0]**: AES suspend
Upon suspend operation, this bitfield of the corresponding AES_SUSPxR register takes the value of one of internal AES registers.

21.7.18 AES register map

Table 130. AES register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	AES_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	WRERR	RDERR	CCF
	Reset value																													0	0	0	0
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	AES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	AES_SUSP0R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 130. AES register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x044	AES_SUSP1R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	AES_SUSP2R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	AES_SUSP3R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	AES_SUSP4R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	AES_SUSP5R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	AES_SUSP6R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	AES_SUSP7R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x3FF	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Refer to *Section 2.4 on page 62* for the register boundary addresses.

22 Public key accelerator (PKA)

22.1 Introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(*p*) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

All needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

22.2 PKA main features

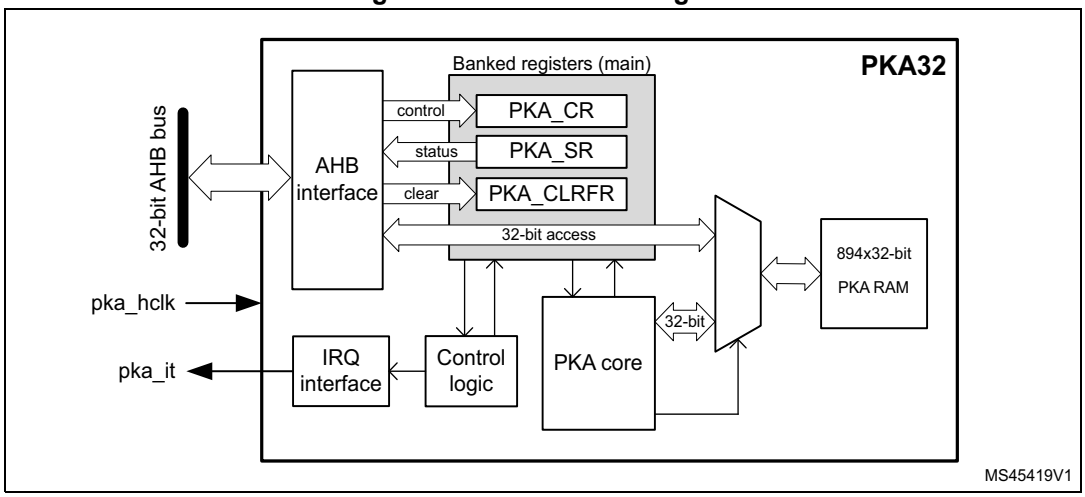
- Acceleration of RSA, DH and ECC over GF(*p*) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA Chinese Remainder Theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check
  - ECDSA signature generation and verification
- Capability to handle operands up to 3136 bits for RSA/DH and 640 bits for ECC.
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication.
- Built-in Montgomery domain inward and outward transformations.
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise, for writes, an AHB bus error is generated, and write accesses are ignored).

22.3 PKA functional description

22.3.1 PKA block diagram

Figure 113 shows the block diagram of the public key accelerator PKA.

Figure 113. PKA block diagram



22.3.2 PKA internal signals

Table 131 lists internal signals available at the IP level, not necessarily available on product bonding pads.

Table 131. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request

22.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. The RAM receives this clock directly, the core is clocked at half the frequency.

When the PKA peripheral reset signal is released PKA RAM is cleared automatically, taking 894 clock cycles. During this time the setting of EN bit in PKA_CR is ignored.

22.3.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 3136 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information is found in *Section 22.5.1: Supported elliptic curves*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 3576 bytes (894 words of 32 bits) called PKA RAM is used for providing initial data to the PKA, and for holding the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 132* and *Table 133*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register.

Table 132. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R2 mod n	*Section 22.4.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 22.4.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 22.4.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 22.4.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 22.4.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x08	001000	Modular inversion A-1 mod n	*Section 22.4.7*
0x0D	001101	Modular reduction A mod n	*Section 22.4.8*
0x09	001001	Arithmetic addition A+B	*Section 22.4.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 22.4.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 22.4.11*
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 22.4.12*
0x07	000111	RSA CRT exponentiation	*Section 22.4.13*

Table 133. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 22.4.14*
0x20	100000	ECC scalar multiplication kP	*Section 22.4.15*
0x22	100010	ECC scalar multiplication kP (fast mode)	
0x24	100100	ECDSA sign	*Section 22.4.16*
0x26	100110	ECDSA verification	*Section 22.4.17*

Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is

supplied before starting the operation. Performance improvement is detailed in *Section 22.5.2: Computation times*.

The operations using fast mode are modular exponentiation and scalar multiplication.

22.3.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions. In particular:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

Specifications of the above functions are given in following publications:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

RSA key pair

For following RSA operations a public key and a private key information are defined as below:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information on above representations refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (*n*, *e*) must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer *m*, with $0 \leq m < n$ and (*m*, *n*) being co-primes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext *m* is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\ (m_1 - m_2) \bmod p$
   d) $m = m_2 + h\ q$
3. Convert the integer message representative *m* to an encoded message EM.
4. Recover message M= DECODE(EM), where DECODE is a decoding method.

Above operations can be accelerated by PKA using *Modular exponentiation* $A^e \bmod n$ if the private key is *d*, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

Elliptic curve selection

For following ECC operations curve parameters are defined as below:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 22.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (i.e. *n* x G = identity element O).

ECDSA message signature generation

ECDSA (Elliptic Curve Digital Signature Algorithm) signature generation function principle is the following: Alice, to sign a message *m* using her private key integer d_A, follows the steps below.

1. Calculate e = HASH(*m*), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order *n*.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1} (z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (*r*, *s*).

Steps 4 to 7 are accelerated by PKA using:

- *ECDSA sign* or
- All of the operations below:
  - *ECC Fp scalar multiplication* k x P
  - *Modular reduction* A mod *n*
  - *Modular inversion* $A^{-1} \bmod n$
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A.

Bob can verify that Q_A is a valid curve point going through the following steps:

1. check that Q_A is not equal to the identity element O
2. check that Q_A is on the agreed curve
3. check that n x Q_A = O.

Then Bob follows the procedure detailed below:

1. verify that r and s are integer in [1, n-1]
2. calculate e = HASH(m), where HASH is the agreed cryptographic hash function
3. let z be the L_n leftmost bits of e
4. calculate w = s^{-1} mod n
5. calculate u_1 = zw mod n and u_2 = rw mod n
6. calculate the curve point (x_1, y_1) = u_1 x G + u_2 x Q_A
7. the signature is valid if r = x_1 (mod n), it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using *ECDSA verification*.

22.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 22.4*.

Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set to “1”, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA_SR.

Using precomputed Montgomery parameters (PKA fast mode)

As explained in *Section 22.3.4*, when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, *Section 22.4.5*). This is know as “fast mode”.

To manage Fast Mode usage the recommended procedure is described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in *Section 22.5.1*.
2. Program in PKA_CR register the PKA in *Montgomery parameter computation* mode (MODE=”0x1”) then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set to “1”, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.
4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in *Section 22.4*.

22.3.7 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the Address Error flag (ADDRERRF) is set in the PKA_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM Error Flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return zero, while writes are ignored.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see *Section 22.6* for details).

ADDRERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

22.4 PKA operating modes

22.4.1 Introduction

The various operations supported by PKA are described in the following subsections, clarifying the associated format of the input data and of the results, both stored in the PKA RAM.

The following information applies to all PKA operations.

- PKA core processes 32-bit words
- Supported operand “Size” are:
  - ROS (RSA operand size): data size is (*rsa_size*/32+1) words, with *rsa_size* equal to the chosen modulus length. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 33 words, or 1056 bits.
  - EOS (ECC operand size): data size is (*ecc_size*/32+1) words, with *ecc_size* equal to the chosen prime modulus length. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 7 words, or 224 bits.

Note: Fractional results for above formulas are rounded up to the nearest integer since PKA core processes 32-bit words.

Note: The maximum ROS is 99 words (3136-bit max exponent size), while the maximum EOS is 21 words (640-bit max operand size).

- The column indicated with Storage in the following tables indicates either the Register PKA_CR, or the address offset within the PKA, located in the PKA RAM area (starting from 0x400). To recover the physical address of an operand the application must add to the indicated offset the base address of the PKA.
- About writing parameters (“IN” direction)
  - When elements are written as input in the PKA memory, an additional word with all bits equal to zero must be added. As an example, when ECC P256 is used and when loading an input (represented on 256 bits or 8 words), an additional word is expected by the PKA and it has to be filled with zeros.
  - About endianess, for example to prepare the operation ECC Fp scalar multiplication, when application writes x_p coordinate for an ECC P256 curve (EOS= 9 words) the least significant bit is to be placed in bit 0 at address offset 0x55C; the most significant bit is to be placed in bit 31 of address offset 0x578. Then, as mentioned above, word 0x00000000 should also be written at address offset 0x57C.
  - Unless indicated otherwise all operands in the tables are integers.
- About PKA outputs (“OUT” direction)
  - Unless specified in the tables PKA does not provide an error output when a wrong parameter is written by the application.

Caution: Validity of all input parameters to the PKA must be checked before issuing any PKA operation. Indeed, the PKA assumes that all input parameters are valid and consistent with each other.

22.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod *n*) used by PKA to convert operands into the Montgomery residue system representation.

Note: This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.

Operation instructions for Montgomery parameter computation are summarized in *Table 134*.

Table 134. Montgomery parameter computation

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x01	PKA_CR	6 bits
	Modulus length	(In bits, 0 ≤ value < 3136bits)	RAM@0x404	32 bits
	Modulus value n	(Odd integer only, $n < 2^{3136}$)	RAM@0xD5C	ROS
OUT	Result: R^2 n	-	RAM@0x594	

22.4.3 Modular addition

Modular addition operation consists in the computation of A + B mod *n*. Operation instructions are summarized in *Table 135*.

Table 135. Modular addition

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0E	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ A < n)	RAM@0x8B4	ROS
	Operand B	(0 ≤ B < n)	RAM@0xA44	
	Modulus value n	(n < 2^{3136})	RAM@0xD5C	
OUT	Result: A+B mod n	(0 ≤ result < n)	RAM@0xBD0	

22.4.4 Modular subtraction

Modular subtraction operation consists in the following computations:

- If A ≥ B result equals A - B mod *n*
- If A < B result equals A + n - B mod *n*

Operation instructions are summarized in *Table 136*.

Table 136. Modular subtraction

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0F	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ A < n)	RAM@0x8B4	ROS
	Operand B	(0 ≤ B < n)	RAM@0xA44	
	Modulus value n	(n < 2^{3136})	RAM@0xD5C	
OUT	Result: A-B mod n	(0 ≤ result < n)	RAM@0xBD0	

22.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication A x B mod *n*

The method to perform above operations are described below. Note that “x” function is this operation, and A, B, C operands are in the natural domain.

1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Let's assume A is an integer in the natural domain
      Compute *r2modn* using *Montgomery parameter computation*
      Result AR= A x *r2modn* mod *n* is A in the Montgomery domain
   b) Let's assume BR is an integer in the Montgomery domain
      Result B = BR x 1 mod *n* is B in the natural domain
      Similarly, above value AR computed in a) can be converted into the natural domain by computing A = AR x 1 mod *n*
2. Simple modular multiplication A x B mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain
   c) Compute AB= AR x B mod *n*. Output is in natural domain
3. Multiple modular multiplication A x B x C mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain
   c) Compute BR = B x *r2modn* mod *n*. Output is in the Montgomery domain
   d) Compute ABR= AR x BR mod *n*. Output is in the Montgomery domain
   e) Compute CR = C x *r2modn* mod *n*. Output is in the Montgomery domain
   f) Compute ABCR= ABR x CR mod *n*. Output is in the Montgomery domain
   g) (optional) Repeat the two steps above if more operands need to be multiplied
   h) Compute ABC= ABCR x 1 mod *n* to retrieve the result in natural domain

Operation instructions for Montgomery multiplication are summarized in *Table 137*.

Table 137. Montgomery multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x10	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ A < n)	RAM@0x8B4	ROS
	Operand B	(0 ≤ B < n)	RAM@0xA44	
	Modulus value n	(Odd integer only, $n < 2^{3136}$)	RAM@0xD5C	
OUT	Result: AxB mod n(1)	-	RAM@0xBD0	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

22.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of A^e mod *n*.

Operation instructions for modular exponentiation are summarized in *Table 138* (normal mode) and in *Table 139* (fast mode). Fast mode usage is explained in *Section 22.3.6*.

Table 138. Modular exponentiation (normal mode)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	32 bits
	Operand length	(in bits, not null)	RAM@0x404	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xA44	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xBD0	
	Modulus value n	(Odd integer only, n < 2^{3136})	RAM@0xD5C	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x724	

Table 139. Modular exponentiation (fast mode)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	32 bits
	Operand length	(in bits, not null)	RAM@0x404	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xA44	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xBD0	
	Modulus value n	(Odd integer only, n < 2^{3136})	RAM@0xD5C	
IN/OUT	Montgomery param R2 mod n	(mandatory)	RAM@0x594	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x724	

22.4.7 Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse A^{-1} mod *n*. If the modulus *n* is prime, for all values of A (1 ≤ A < *n*) modular inversion output is valid. If the modulus *n* is not prime, A has an inverse only if the largest common divisor between A and *n* is 1.

If the operand A is a divisor of the modulus *n,* the result is a multiple of a factor of *n*.

Operation instructions for modular inversion are summarized in *Table 140*.

Table 140. Modular inversion

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x08	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ A < n)	RAM@0x8B4	ROS
	Modulus value n	(Odd integer only, n < 2^{3136})	RAM@0xA44	
OUT	Result: A^{-1} mod n	0 < result < n	RAM@0xBD0	

22.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of A divided by n. Operation instructions are summarized in *Table 141*.

Table 141. Modular reduction

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0D	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x400	32 bits
	Modulus length	(In bits, 8 < value < 3136)	RAM@0x404	
	Operand A	($0 \le A < 2n < 2^{3136}$)	RAM@0x8B4	ROS
	Modulus value n	(Odd integer only, $n < 2^{3136}$)	RAM@0xA44	
OUT	Result A mod n	(0 < result < n)	RAM@0xBD0	

22.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of A + B. Operation instructions are summarized in *Table 142*.

Table 142. Arithmetic addition

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x09	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	($0 \le < A < 2^{M}$)	RAM@0x8B4	ROS
	Operand B	($0 \le < B < 2^{M}$)	RAM@0xA44	
OUT	Result: A+B	($0 \le$ result $< 2^{M+1}$)	RAM@0xBD0	ROS + 1

22.4.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If A ≥ B result equals A - B
- If A < B and M/32 residue is >0 result equals A + $2^{int(M/32)*32+1}$ - B
- If A < B and M/32 residue is 0 result equals A + $2^{int(M/32)*32}$ - B

Operation instructions are summarized in *Table 143*.

Table 143. Arithmetic subtraction

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0A	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	($0 \le < A < 2^{M}$)	RAM@0x8B4	ROS
	Operand B	($0 \le < B < 2^{M}$)	RAM@0xA44	
OUT	Result: A-B	($0 \le$ result $< 2^{M}$)	RAM@0xBD0	

22.4.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in *Table 144*.

Table 144. Arithmetic multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0B	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	($0 \leq < A < 2^M$)	RAM@0x8B4	ROS
	Operand B	($0 \leq < B < 2^M$)	RAM@0xA44	
OUT	Result: AxB	($0 \leq$ result $< 2^M$)	RAM@0xBD0	2xROS

22.4.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:

- If A=B then result=0x0
- If A>B then result=0x1
- If A<B then result=0x2

Operation instructions for arithmetic comparison are summarized in *Table 145*.

Table 145. Arithmetic comparison

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0C	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	($0 \leq < A < 2^M$)	RAM@0x8B4	ROS
	Operand B	($0 \leq < B < 2^M$)	RAM@0xA44	
OUT	Result A=B or A>B or A<B	0x0, 0x1 or 0x02	RAM@0xBD0	32 bits

22.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries like OpenSSL RSA use the following optimization for decryption and signing based on the Chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p-1)$
- $d_Q = d \bmod (q-1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation $m = A^d \pmod{pq}$ more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod p$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq$

Operation instructions for computing CRT exponentiation $A^d \bmod pq$ are summarized in *Table 146*.

Table 146. CRT exponentiation

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x404	32 bits
IN	Operand d_P	$(0 \le d_P < 2^{M/2})$	RAM@0x65C	ROS/2
	Operand d_Q	$(0 \le d_Q < 2^{M/2})$	RAM@0xBD0	
	Operand q_{inv}	$(0 \le q_{inv} < 2^{M/2})$	RAM@0x7EC	
	Prime p(1)	$(0 \le p < 2^{M/2})$	RAM@0x97C	
	Prime q(1)	$(0 \le q < 2^{M/2})$	RAM@0xD5C	
IN	Operand A	$(0 \le A < 2^{M/2})$	RAM@0xEEC	ROS
OUT	Result: $A^d \bmod pq$	$(0 \le result < pq)$	RAM@0x724	

1. Must be different from 2.

22.4.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point P (x, y) satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \bmod p$, where a and b are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in *Table 147*.

Table 147. Point on elliptic curve Fp check

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x28	PKA_CR	6 bits
	Modulus length	(In bits, not null, 8 < value < 640)	RAM@0x404	32 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x408	
	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x40C	EOS
	Curve coefficient *b*	(|b| < p)	RAM@0x7FC	
	Curve modulus value *p*	(Odd integer prime, 0 < p < 2640)	RAM@0x460	
	Point P coordinate *x*	(x < *p*)	RAM@0x55C	
	Point P coordinate *y*	(y < *p*)	RAM@0x5B0	
OUT	Result: P on curve	0x0: point on curve Not 0x0: point not on curve	RAM@0x400	32 bits

22.4.15 ECC Fp scalar multiplication

This operation consists in the computation of a k x P (x_P, y_P), where P is a point on a curve over prime fields and “x” is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in *Table 148* (normal mode) and *Table 149* (fast mode). Fast mode usage is explained in *Section 22.3.6*.

Table 148. ECC Fp scalar multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x20	PKA_CR	6 bits
IN	Scalar multiplier k length	(In bits, not null, 8 < value < 640)	RAM@0x400	32 bits
	Modulus length	(In bits, not null, 8 < value < 640)	RAM@0x404	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x408	
IN	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x40C	EOS
	Curve modulus value *p*	(Odd integer prime, $0 < p < 2^{640}$)	RAM@0x460	
	Scalar multiplier k	($0 \leq k < 2^{640}$)	RAM@0x508	
	Point P coordinate x_P	(x < *p*)	RAM@0x55C	
	Point P coordinate y_P	(y < *p*)	RAM@0x5B0	
OUT	Result: k x P coordinate *x*	(result < *p*)	RAM@0x55C	
	Result: k x P coordinate *y*	(result < *p*)	RAM@0x5B0	

Table 149. ECC Fp scalar multiplication (Fast Mode)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x22	PKA_CR	6 bits
IN	Scalar multiplier k length	(In bits, not null, 8 < value < 640)	RAM@0x400	32 bits
	Modulus length	(In bits, not null, 8 < value < 640)	RAM@0x404	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x408	
IN	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x40C	EOS
	Curve modulus value *p*	(Odd integer prime, $0 < p < 2^{640}$)	RAM@0x460	
	Scalar multiplier k	($0 \leq k < 2^{640}$)	RAM@0x508	
	Point P coordinate x_P	(x < *p*)	RAM@0x55C	
	Point P coordinate y_P	(y < *p*)	RAM@0x5B0	
IN	Montgomery parameter R^2 mod *p*	(mandatory)	RAM@0x4B4	
OUT	Result: k x P coordinate *x*	(result < *p*)	RAM@0x55C	
	Result: k x P coordinate *y*	(result < *p*)	RAM@0x5B0	

When performing this operation following special cases should be noted:

- For k = 0, this function returns a point at infinity, that is (0, 0) if curve parameter b is nonzero and (0, 1) otherwise. For k different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation should be carried out.
- For k < 0 (i.e. a negative scalar multiplication is required) multiplier absolute value k = |-k| should be provided to the PKA. After the computation completion, the formula -P = (x, -y) can be used to compute the y coordinate of the effective final result (the x coordinate remains the same).

22.4.16 ECDSA sign

ECDSA signing operation (outlined in *Section 22.3.5*) is summarized in *Table 150* (input parameters) and in *Table 151* (output parameters).

The application should check if the output error is equal to zero, if it is different from zero a new k should be generated and the ECDSA sign operation should be repeated.

Table 150. ECDSA sign - Inputs

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x24	PKA_CR	6 bits
	Curve prime order n length	(in bits, not null)	RAM@0x400	32 bits
	Curve modulus p length	(in bits, 8 < value < 640)	RAM@0x404	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x408	
	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x40C	EOS
	Curve modulus value *p*	(Odd integer prime, $0 < p < 2^{640}$)	RAM@0x460	
	Integer k(1)	($0 \leq k < 2^{640}$)	RAM@0x508	
	Curve base point G coordinate x	(x < *p*)	RAM@0x55C	
	Curve base point G coordinate y	(y < *p*)	RAM@0x5B0	
	Hash of message z	(z < 2M)	RAM@0xDE8	
	Private key d	(positive integer)	RAM@0xE3C	
	Curve prime order n	(integer prime)	RAM@0xE94	

1. This integer is usually a cryptographically secure random number, but in some cases k could be deterministically generated.

Table 151. ECDSA sign - Outputs

Parameters with direction		Value (Note)	Storage	Size
OUT	Signature part *r*	(0 < *r* < n)	RAM@0x700	EOS
	Signature part *s*	(0 < *s* < n)	RAM@0x754	
ERROR	Result of signature	– 0x0: no error – 0x1: signature part r is equal to 0 – 0x2: signature part s is equal to 0	RAM@0xEE8	32 bits

Note: If error output is different from zero the content of the PKA memory should be cleared to avoid leaking information about the private key.

Extended ECDSA support

PKA also supports Extended ECDSA signature, for which the inputs and the outputs have the same ECDSA signature (*Table 150* and *Table 151*, respectively), with the addition of the coordinates of the point kG. This extra output is defined in *Table 152*.

Table 152. Extended ECDSA sign (extra outputs)

Parameters with direction		Value (Note)	Storage	Size
OUT	Curve point kG coordinate x_1	($0 \le x_1 < p$)	RAM@0x103C	EOS
	Curve point kG coordinate y_1	($0 \le y_1 < p$)	RAM@0x1090	

22.4.17 ECDSA verification

ECDSA verification operation (outlined in *Section 22.3.5*) is summarized in *Table 153* (input parameters) and *Table 154* (output parameters).

The epplication should check if the output error is equal to zero, if it is different from zero, the signature is not verified.

Table 153. ECDSA verification (inputs)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x26	PKA_CR	6 bits
	Curve prime order n length	(In bits, not null)	RAM@0x404	32 bits
	Curve modulus p length	(In bits, not null, 8 < value < 640)	RAM@0x4B4	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x45C	
	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x460	EOS
	Curve modulus value *p*	(Odd integer prime, $0 < p < 2^{640}$)	RAM@0x4B8	
	Curve base point G coordinate x	($x < p$)	RAM@0x5E8	
	Curve base point G coordinate y	($y < p$)	RAM@0x63C	
	Public-key curve point Q coordinate x_Q	($x_Q < p$)	RAM@0xF40	
	Public-key curve point Q coordinate y_Q	($y_Q < p$)	RAM@0xF94	
	Signature part *r*	($0 < r < n$)	RAM@0x1098	
	Signature part *s*	($0 < s < n$)	RAM@0xA44	
	Hash of message *z*	($z < 2^M$)	RAM@0xFE8	
	Curve prime order *n*	(integer prime)	RAM@0xD5C	

Table 154. ECDSA verification (outputs)

Parameters with direction		Value (Note)	Storage	Size
OUT	Result: ECDSA verify	0x0: valid signature Not 0x0: invalid signature	RAM@0x5B0	32 bits

22.5 Example of configurations and processing times

22.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curvescan be described with a short Weierstrass equation $y^2 = x^3 + ax + b \pmod{p}$.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field GF(*p*)
- the (usually prime) integer *n*, the order of the group generated by G, defined below
- the base point of the curve G, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. **a defined as *p*-|*a*|** in the finite field GF(*p*), for example **p-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 155 summarizes the family of curves supported by PKA for ECC operations.

Table 155. Family of supported curves for ECC operations

Curve name	Standard	Reference
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4
P-224		
P-256		
P-384		
P-521		

Table 155. Family of supported curves for ECC operations (continued)

Curve name	Standard	Reference	
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

22.5.2 Computation times

The following tables summarize the PKA computation times, expressed in clock cycles.

Table 156. Modular exponentiation computation times

Exponent length (in bits)	Mode	Modulus length (in bits)		
		1024	2048	3072
3	Normal	304000	814000	1728000
	Fast	46000	164000	356000
17	Normal	326000	896000	1910000
	Fast	68000	246000	534000
$2^{16} + 1$	Normal	416000	1222000	2616000
	Fast	158000	572000	1244000
1024	Normal	11664000	-	-
	Fast	11280000	-	-
	CRT(1)	3546000	-	-
2048	Normal	-	83834000	-
	Fast	-	82046000	-
	CRT(1)	-	23468000	-
3072	Normal	-	-	274954000
	Fast	-	-	273522000
	CRT(1)	-	-	73378000

1. CRT stands for chinese remainder theorem optimization (MODE bitfield = 0x07).

Table 157. ECC scalar multiplication computation times(1)

Mode	Modulus length (in bits)						
	160	192	256	320	384	512	521
Normal	1634000	2500000	4924000	8508000	13642000	28890000	33160000
Fast	1630000	2494000	4916000	8494000	13614000	28842000	33158000

1. These times depend on the number of “1”s included in the scalar parameter.

Table 158. ECDSA signature average computation times(1) (2)

Modulus length (in bits)						
160	192	256	320	384	512	521
1760000	2664000	5249000	9016000	14596000	30618000	35540000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 159. ECDSA verification average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**521**
3500000	5350000	10498000	18126000	29118000	61346000	71588000

Table 160. Point on elliptic curve Fp check average computation times

Modulus length (in bits)					
160	**192**	**256**	**320**	**384**	**512**
10800	14200	20400	31000	49600	82400

Table 161. Montgomery parameters average computation times(1)

Modulus length (in bits)									
160	**192**	**256**	**320**	**384**	**512**	**521**	**1024**	**2048**	**3072**
4518	7846	11848	14902	21682	35012	64000	119536	466146	1104642

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

22.6 PKA interrupts

There are three individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. access to unmapped address (ADDRERRF), see *Section 22.3.7*
2. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 22.3.7*
3. PKA end of operation (PROCENDF)

The three interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 162 gives a summary of the available features.

Table 162. PKA interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
PKA	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit

22.7 PKA registers

22.7.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.
											rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
		rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code
000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
100000: Montgomery parameter computation then ECC scalar multiplication
100010: ECC scalar multiplication only (Montgomery parameter must be loaded first)
100100: ECDSA sign
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
Others: Reserved

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **START**: start the operation

Writing 1 to this bit starts the operation which is selected by MODE[5:0], using the operands and data already written to the PKA RAM. This bit is always read as 0.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable.

0: Disable PKA

1: Enable PKA

Note: When EN=0 PKA RAM can still be accessed by the application.

22.7.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDR ERRF	RAM ERRF	Res.	PROC ENDF	BUSY
											r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADDRERRF**: Address error flag

0: No Address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA End of Operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: PKA operation is in progress

This bit is set to 1 whenever START bit in the PKA_CR is set. It is automatically cleared when the computation is complete, meaning that PKA RAM can be safely accessed and a new operation can be started.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode the peripheral is busy for a couple of cycles, then it aborts automatically the operation and go back to ready (BUSY bit is set to 0).

Bits 15:0 Reserved, must be kept at reset value.

22.7.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDR ERRFC	RAM ERRFC	Res.	PROC ENDFC	Res.
											w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADDRERRFC**: Clear Address error flag
- 0: No action
- 1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
- 0: No action
- 1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA End of Operation flag
- 0: No action
- 1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

22.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 3576 bytes (max word offset: 0x11F4).

22.7.5 PKA register map

Table 163. PKA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value												0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0		0	0																
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0		0																

Refer to *Section 2.4 on page 62* for the register boundary addresses.

23 Advanced-control timer (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

23.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

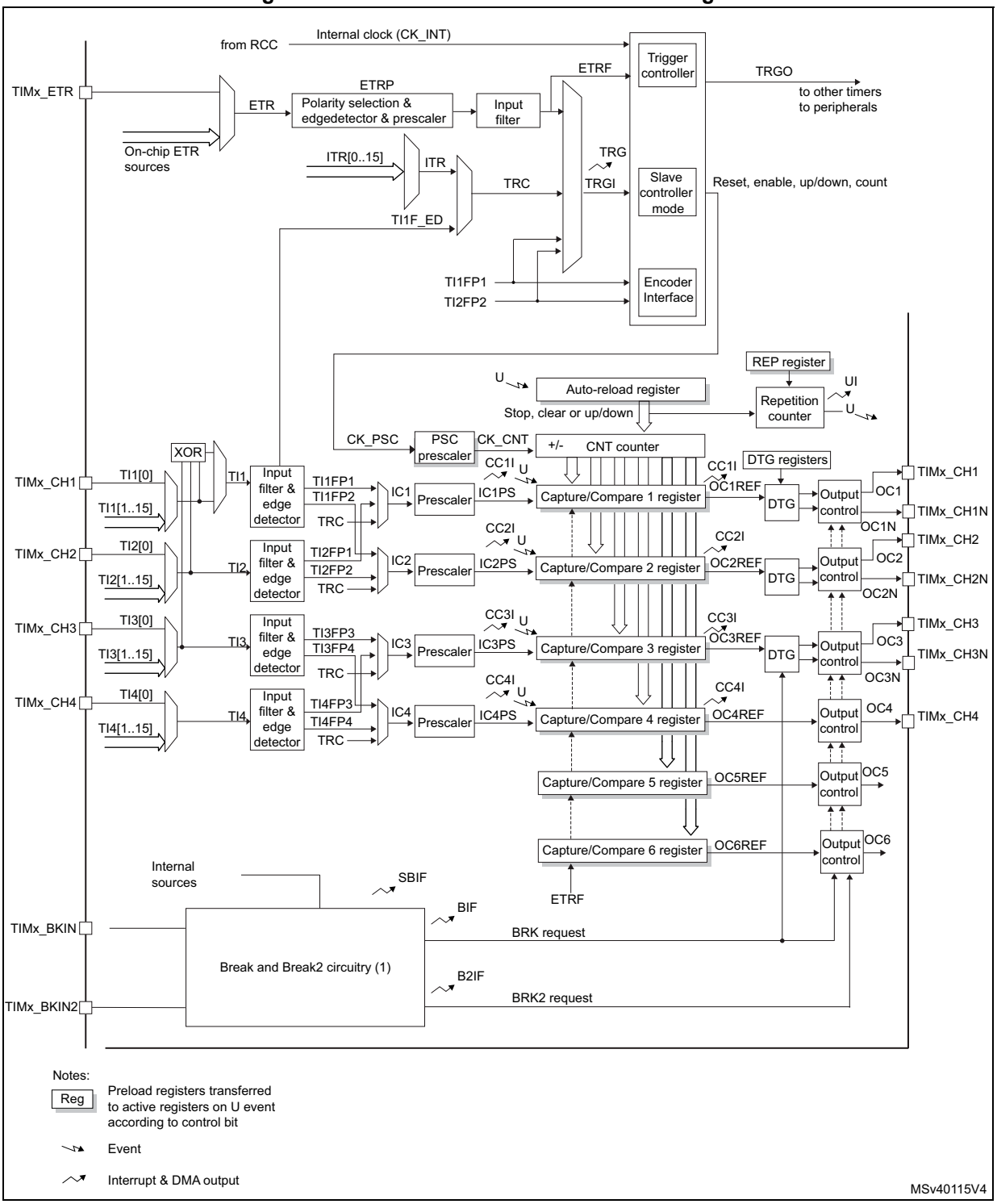
The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 23.3.26: Timer synchronization*.

23.2 TIM1 main features

TIM1 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 114. Advanced-control timer block diagram



1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 6.2.10: Clock security system on HSE32 (CSS)*
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M4 LOCKUP (Hardfault) output.
   - COMPx output, x = 1,2.

23.3 TIM1 functional description

23.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 115 and *Figure 116* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 115. Counter timing diagram with prescaler division change from 1 to 2

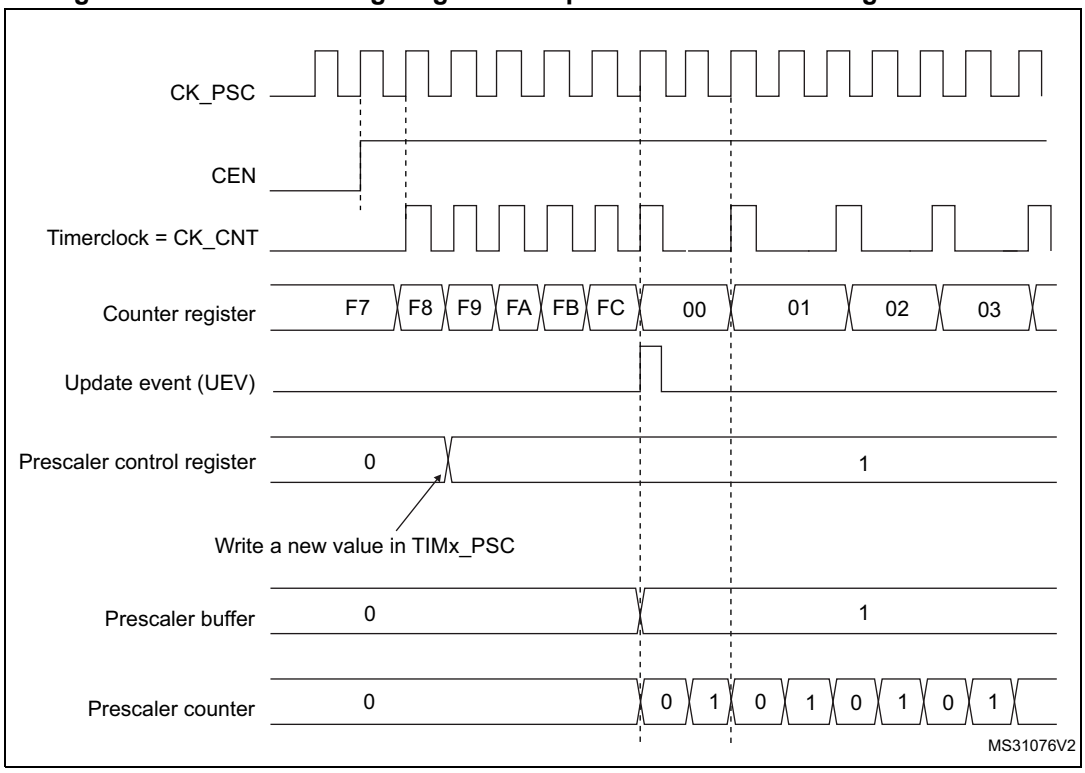
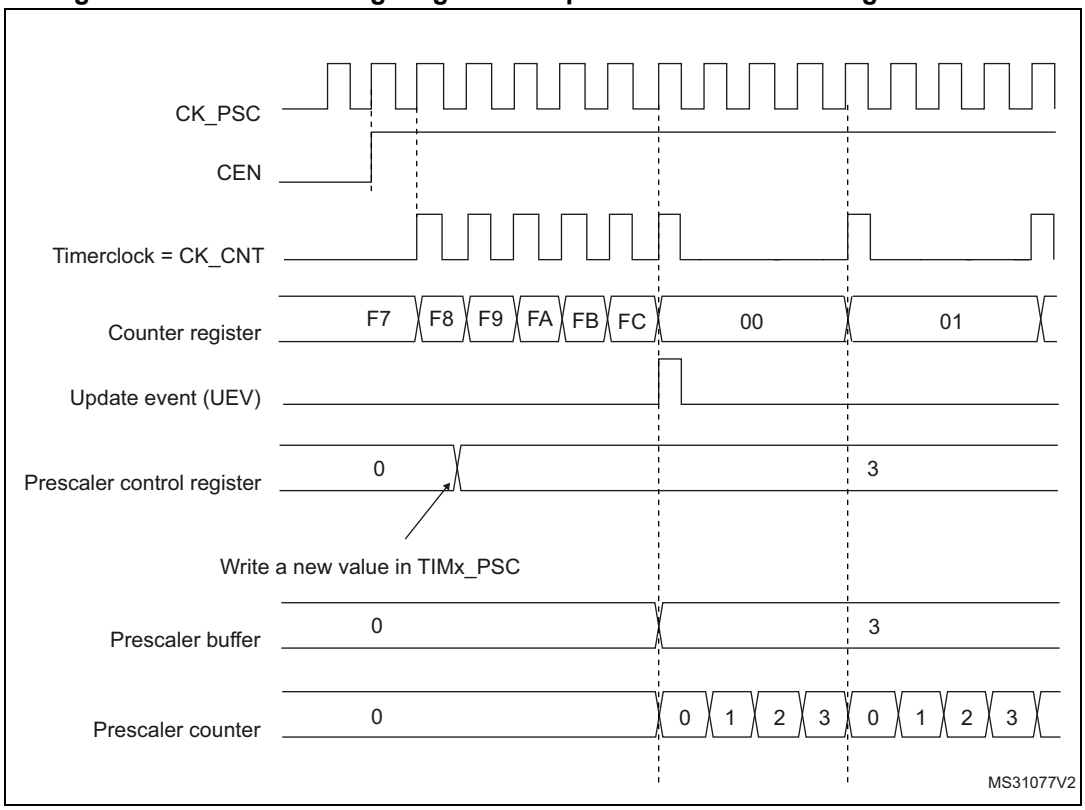


Figure 116. Counter timing diagram with prescaler division change from 1 to 4



23.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 117. Counter timing diagram, internal clock divided by 1

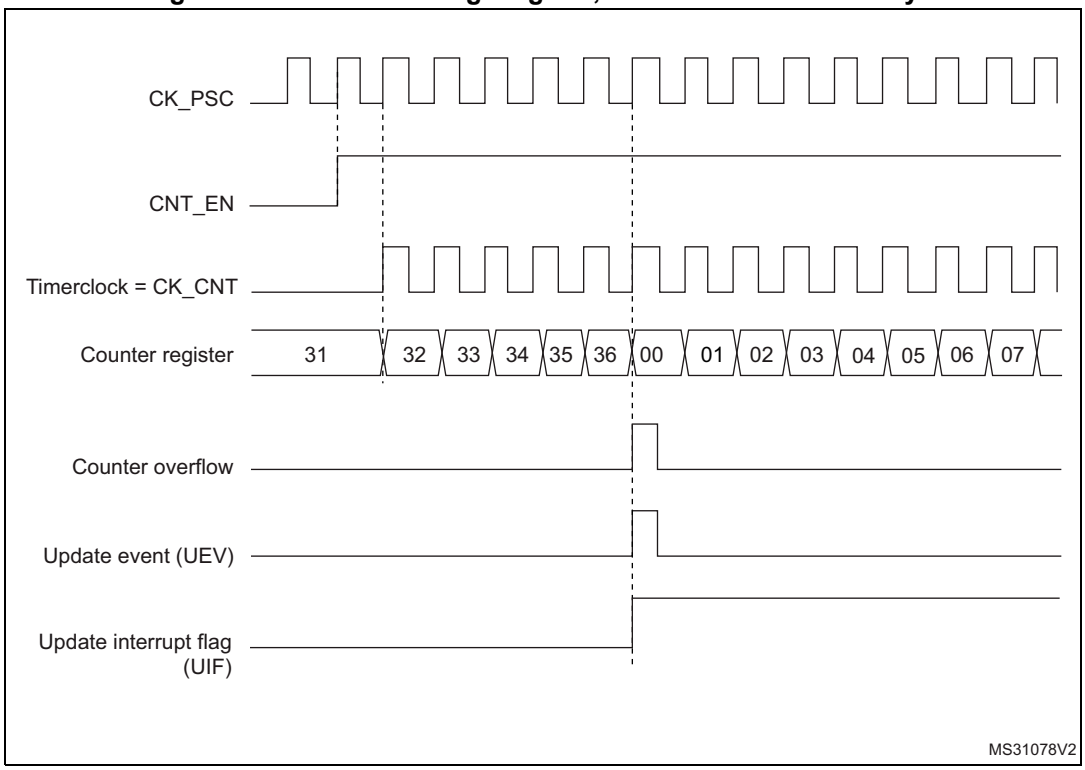


Figure 118. Counter timing diagram, internal clock divided by 2

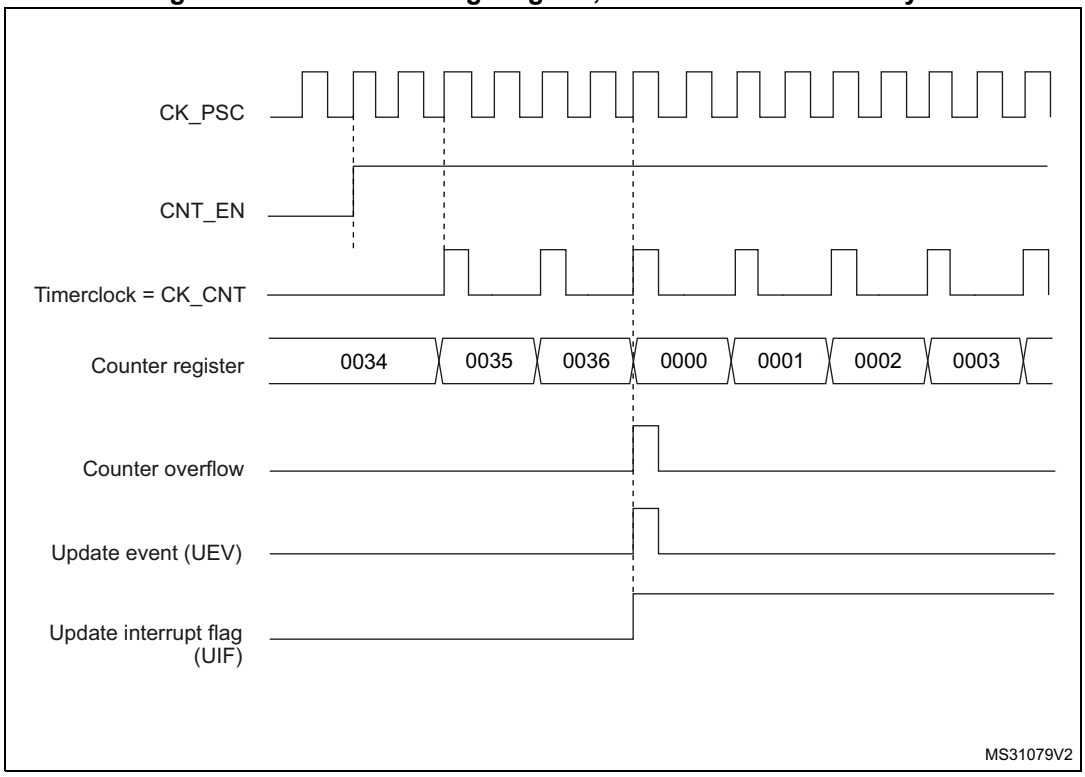


Figure 119. Counter timing diagram, internal clock divided by 4

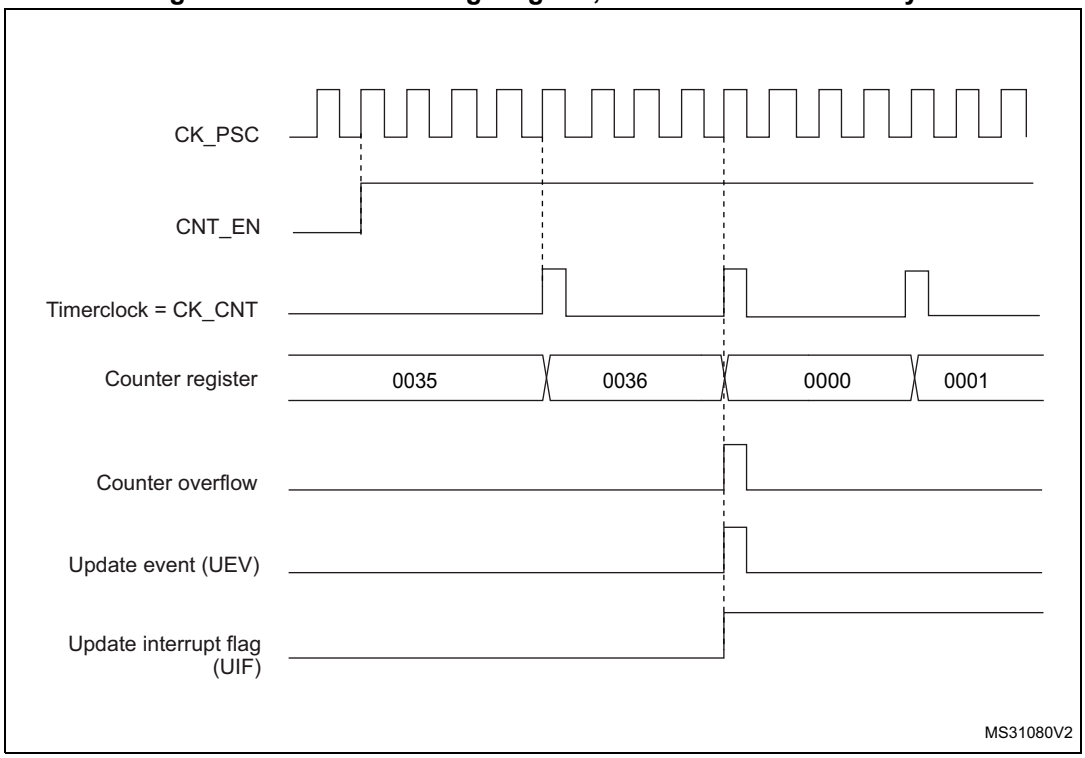


Figure 120. Counter timing diagram, internal clock divided by N

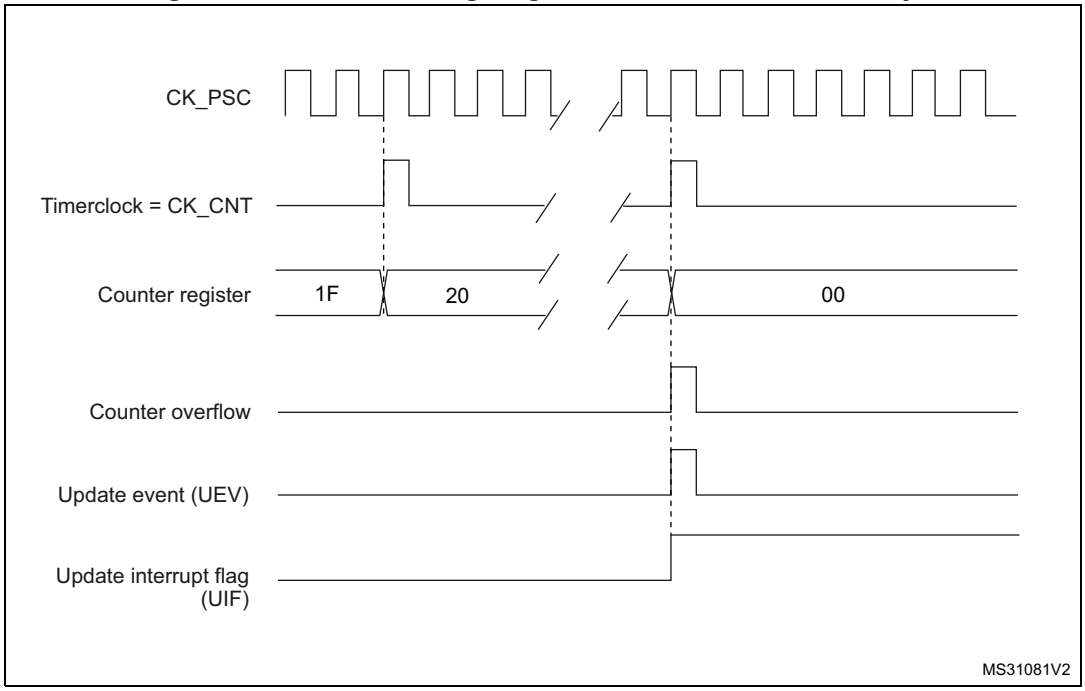


Figure 121. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

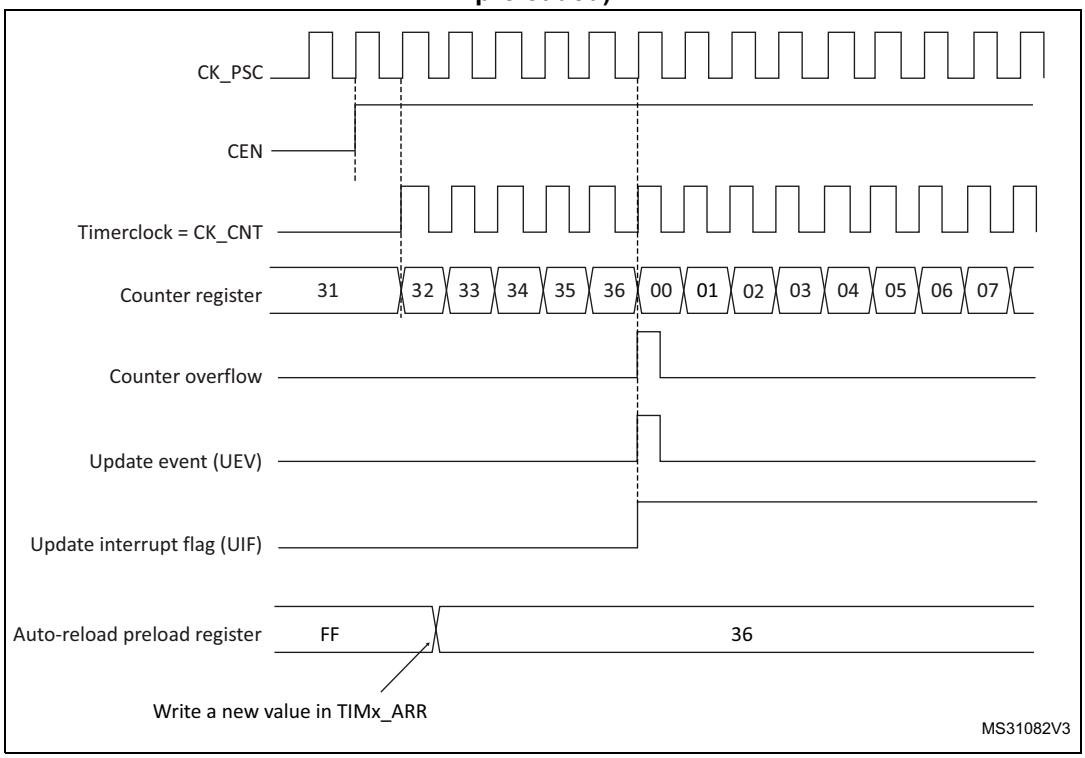
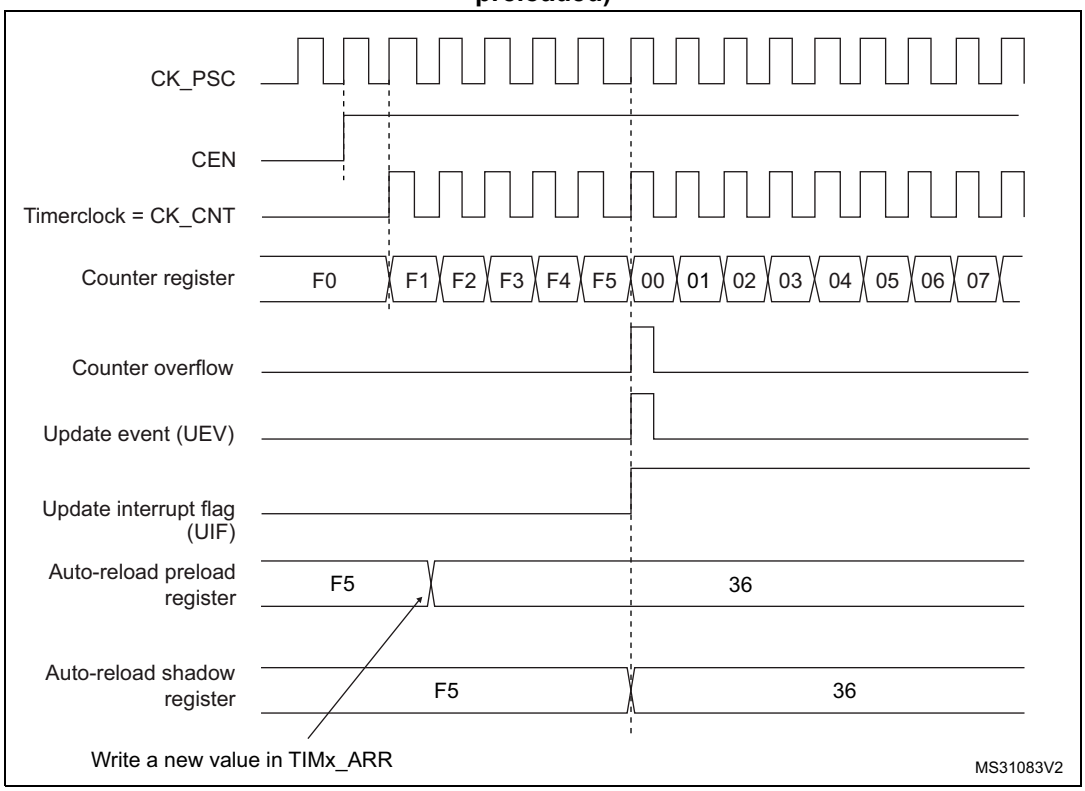


Figure 122. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 123. Counter timing diagram, internal clock divided by 1

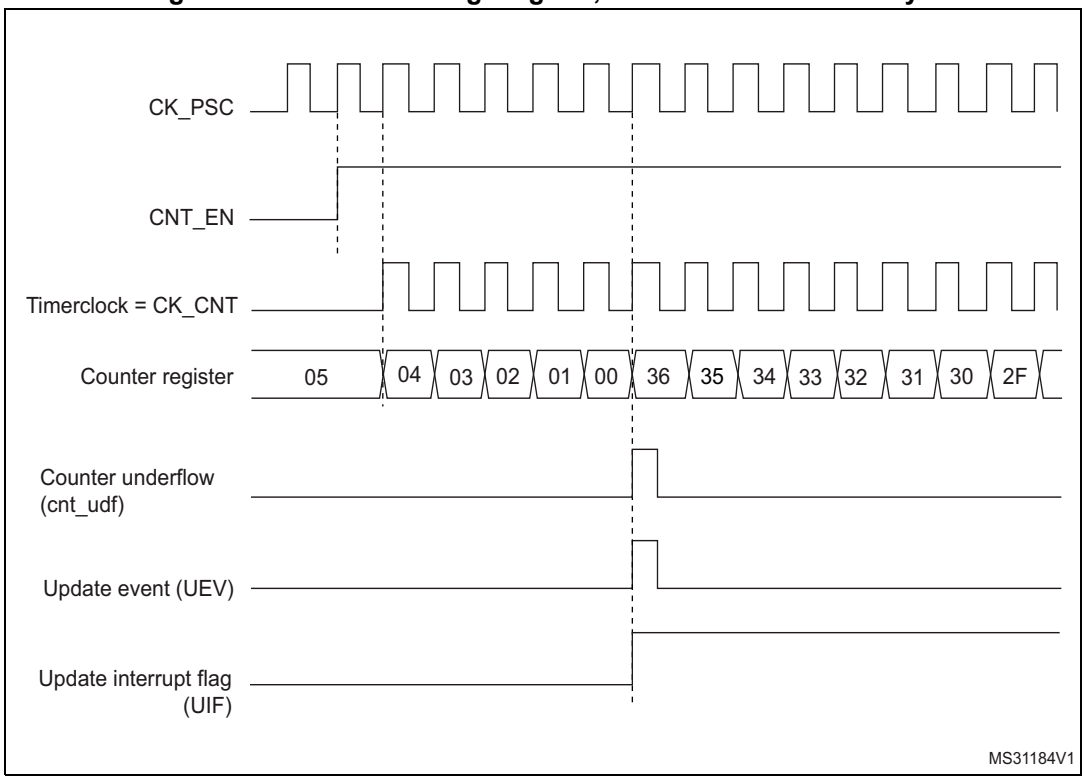


Figure 124. Counter timing diagram, internal clock divided by 2

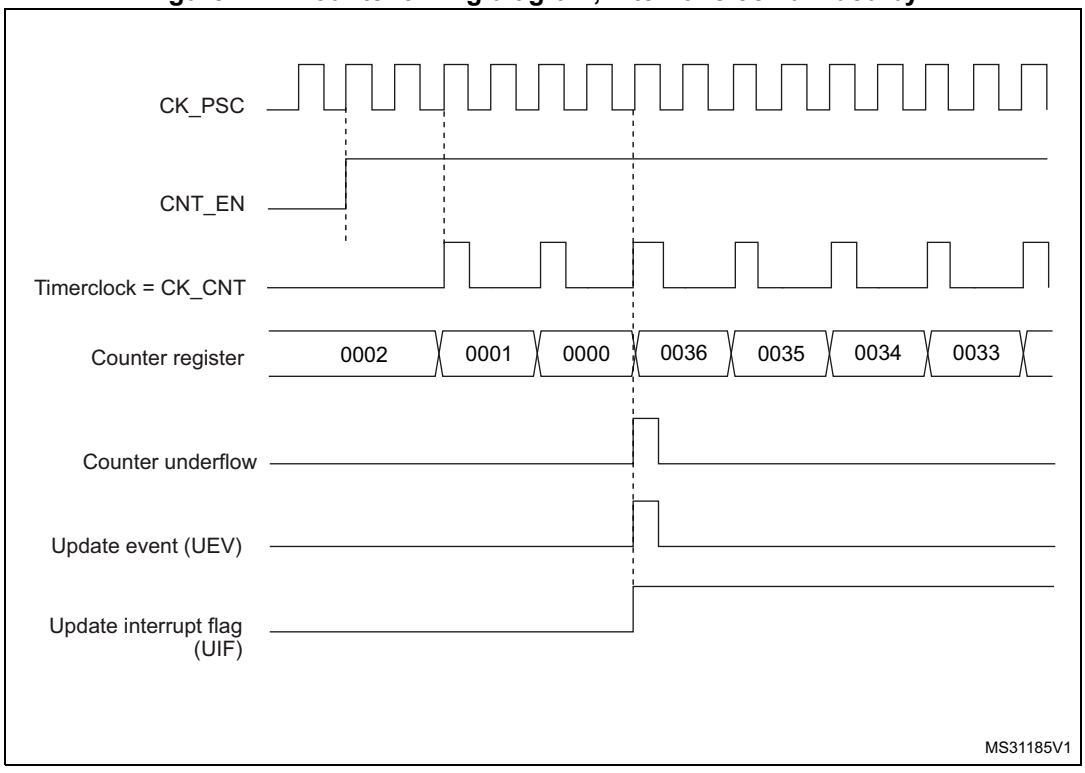


Figure 125. Counter timing diagram, internal clock divided by 4

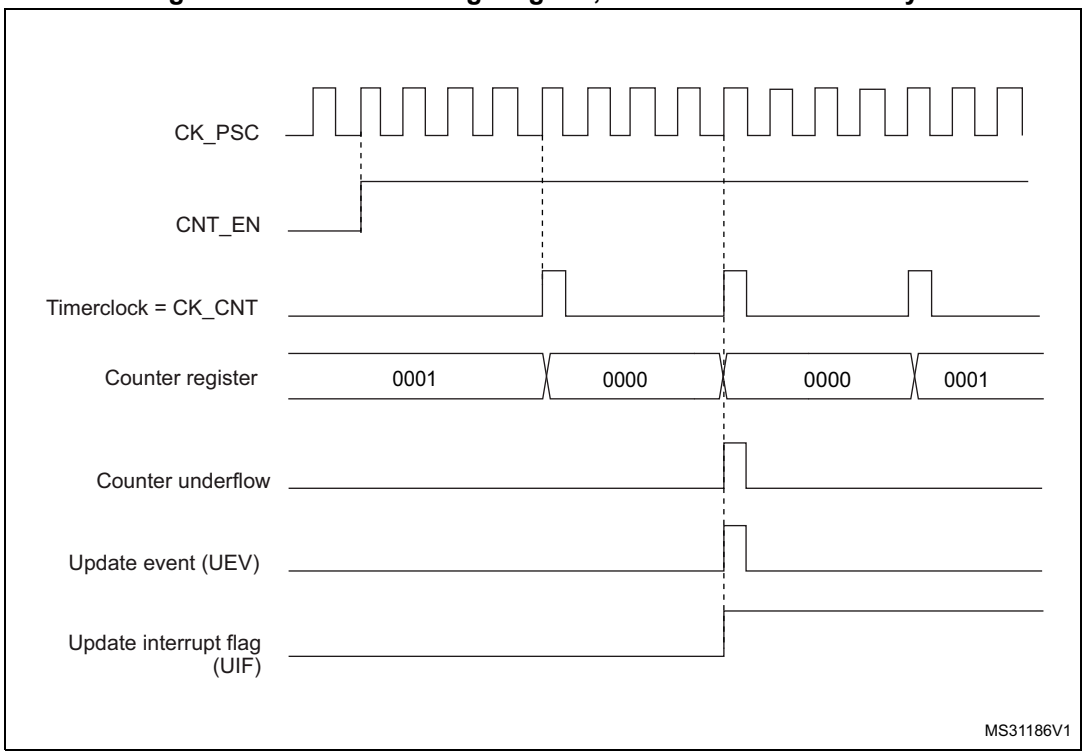


Figure 126. Counter timing diagram, internal clock divided by N

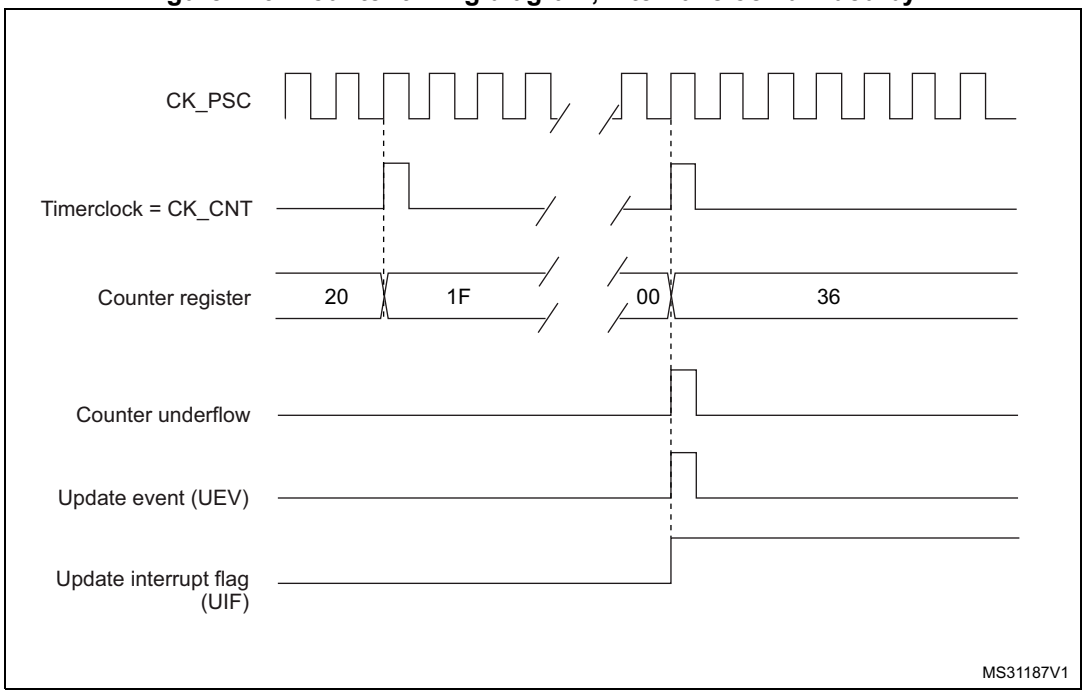
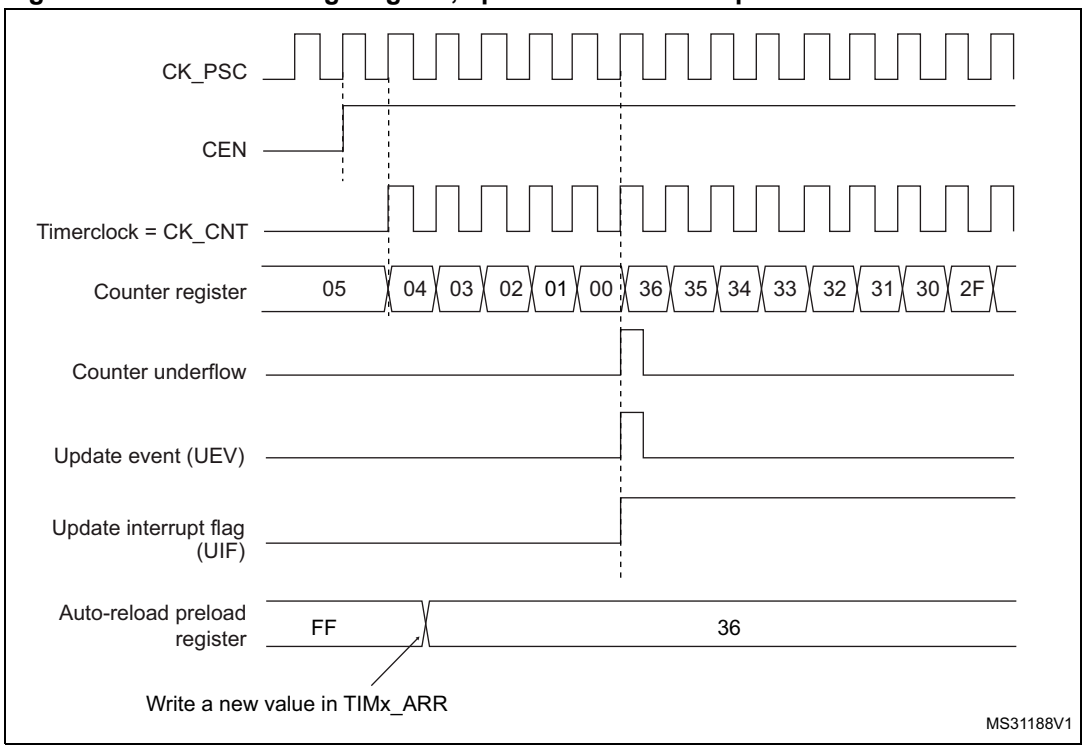


Figure 127. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or

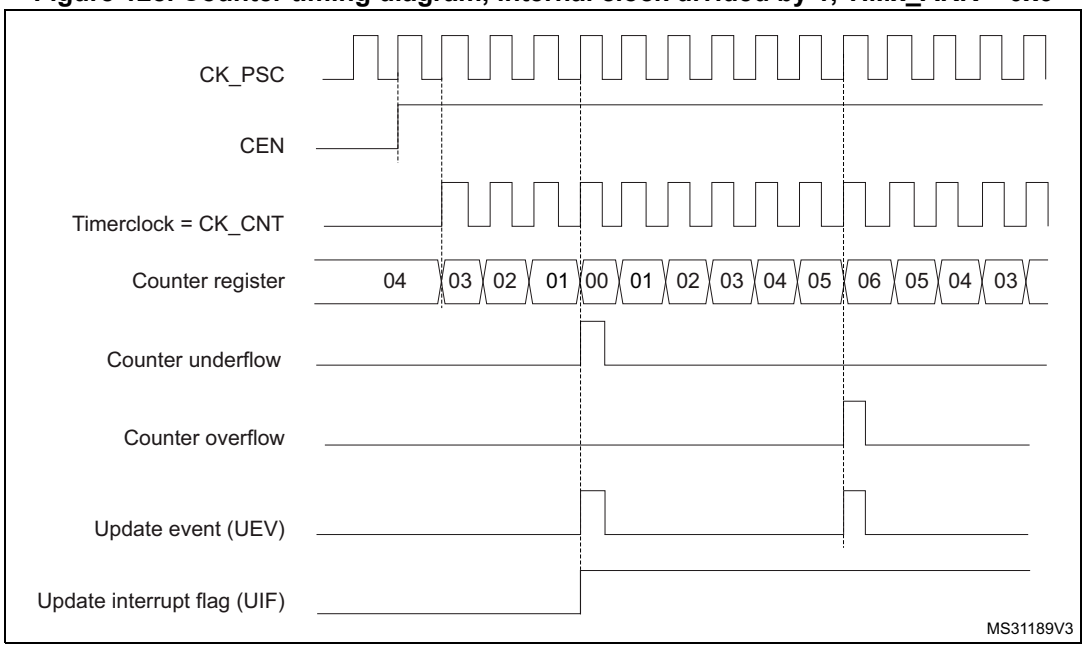
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 128. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 23.4: TIM1 registers*).

Figure 129. Counter timing diagram, internal clock divided by 2

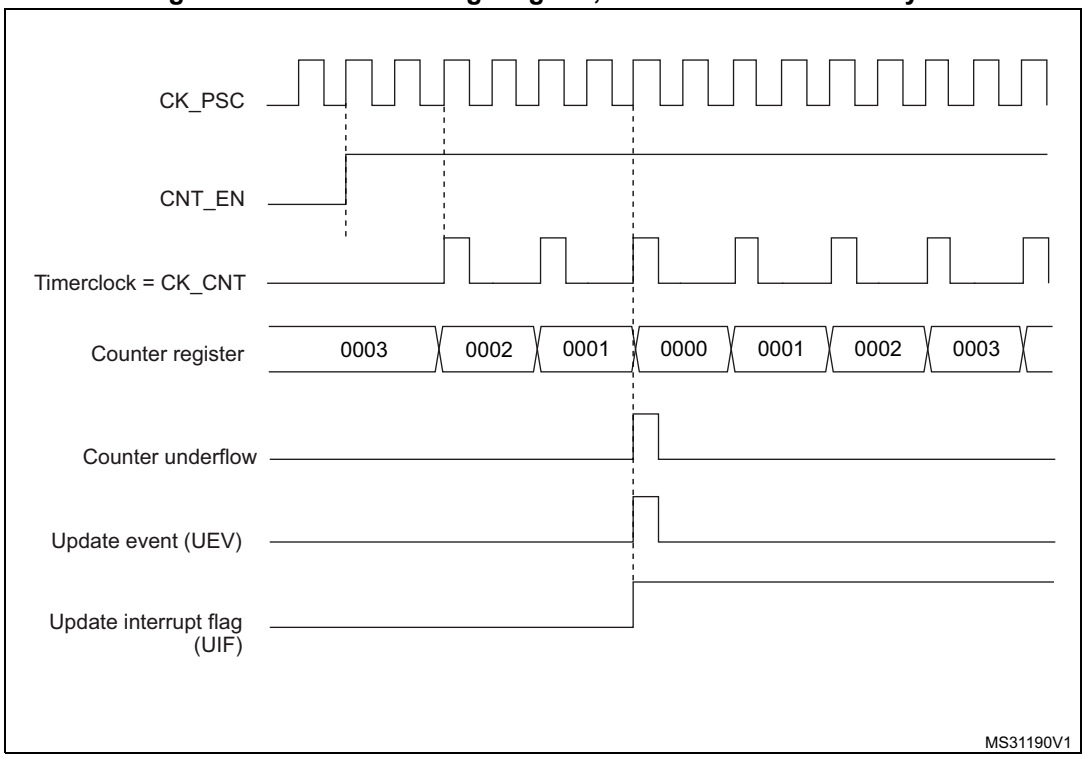


Figure 130. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

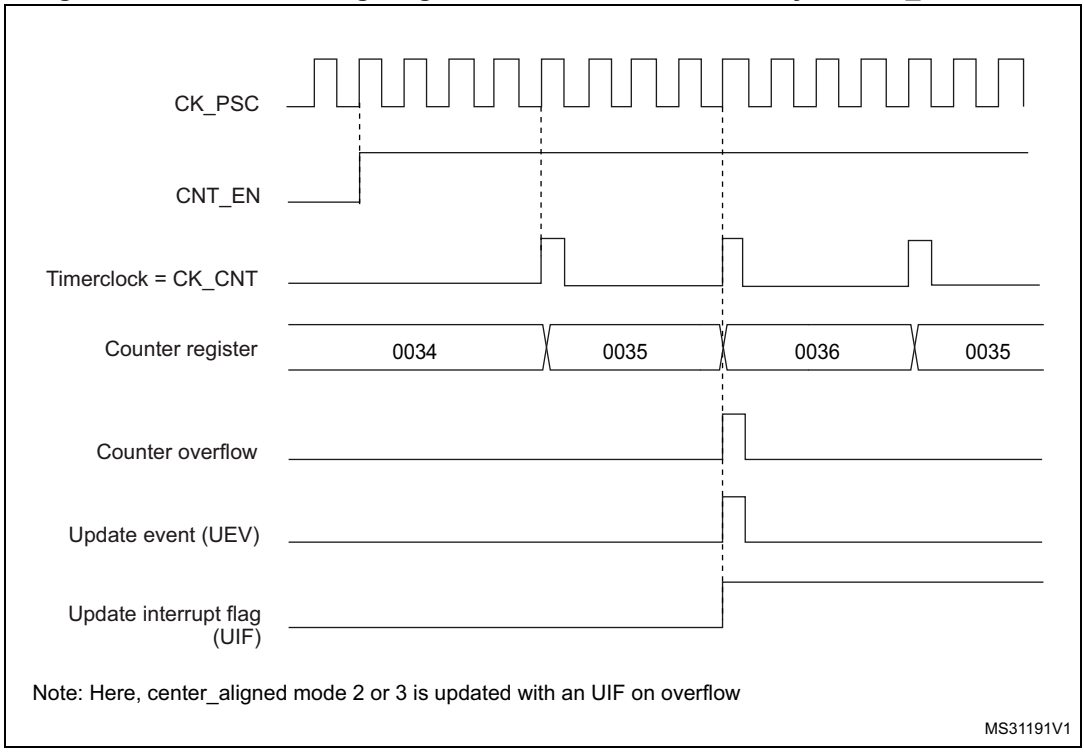


Figure 131. Counter timing diagram, internal clock divided by N

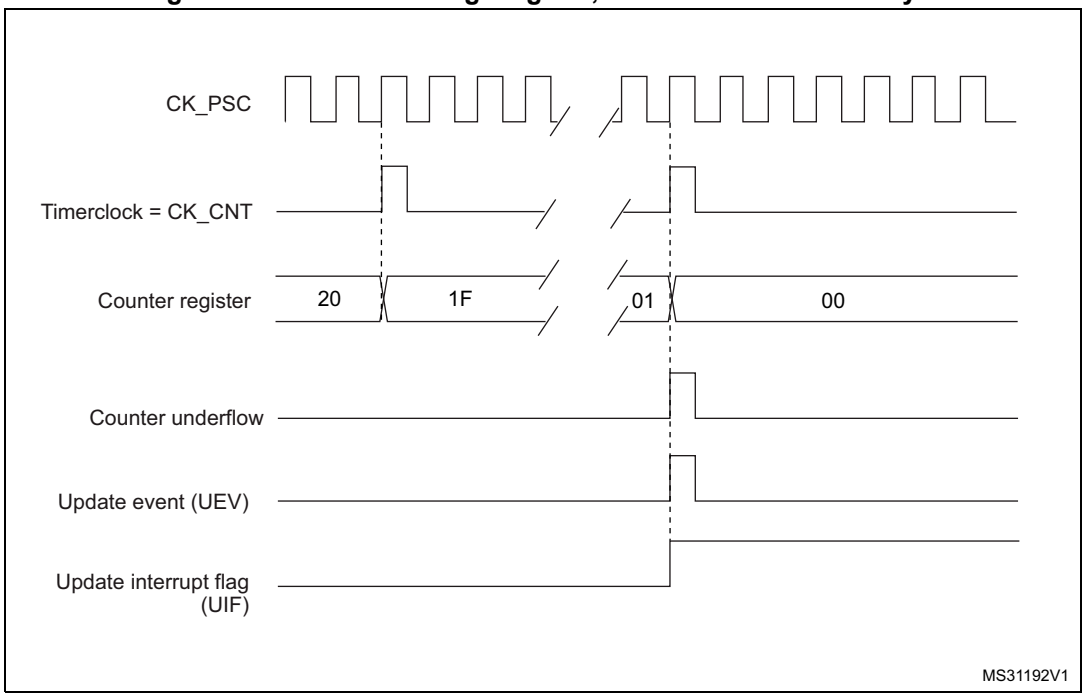


Figure 132. Counter timing diagram, update event with ARPE=1 (counter underflow)

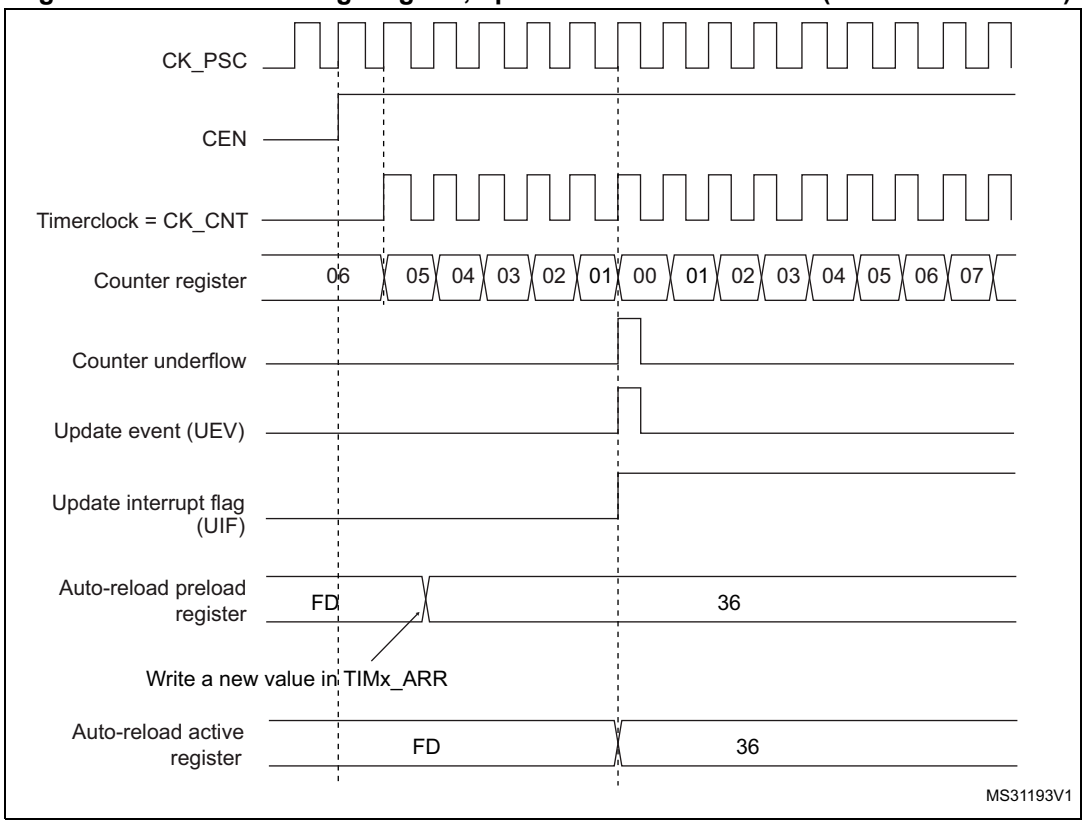
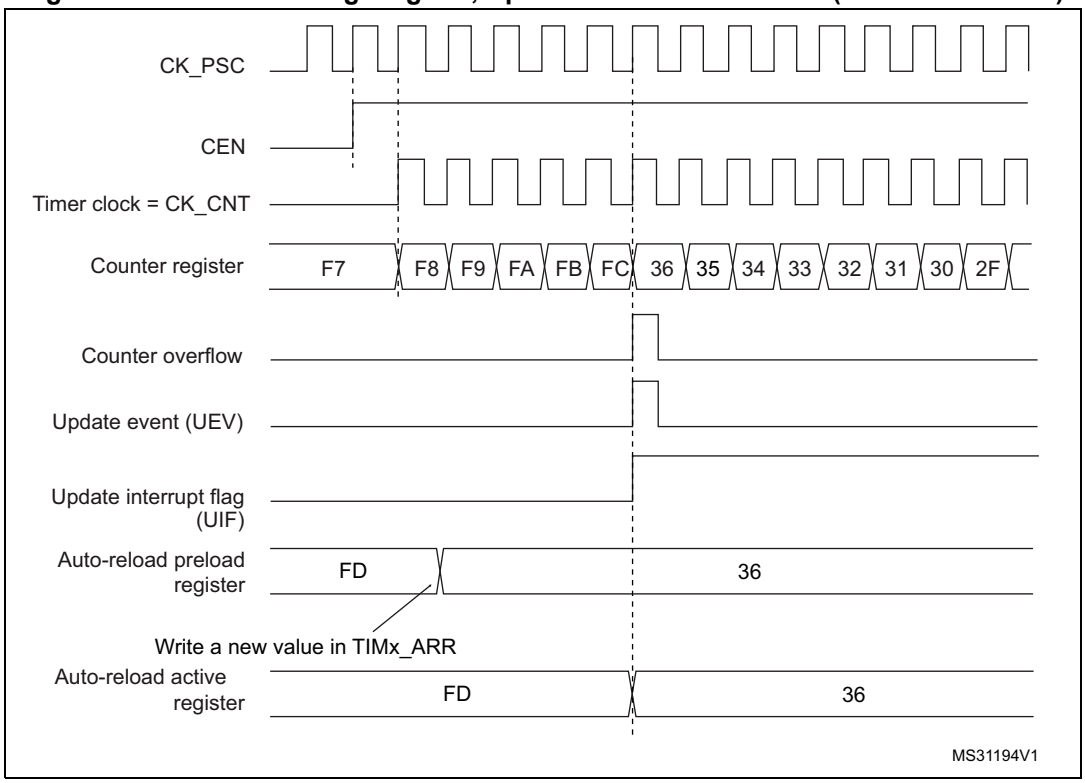


Figure 133. Counter timing diagram, Update event with ARPE=1 (counter overflow)



23.3.3 Repetition counter

Section 23.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

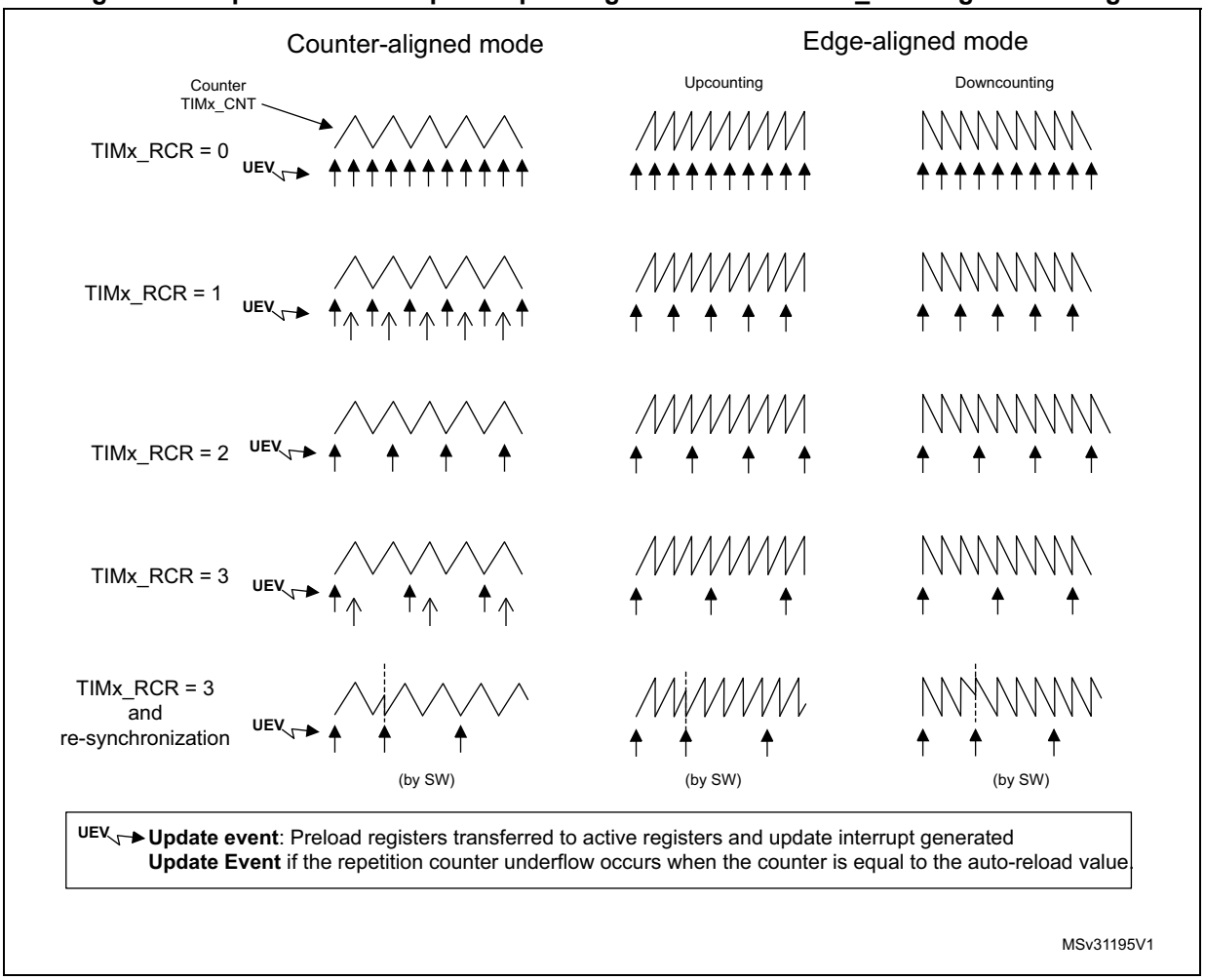
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 134*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 134. Update rate examples depending on mode and TIMx_RCR register settings



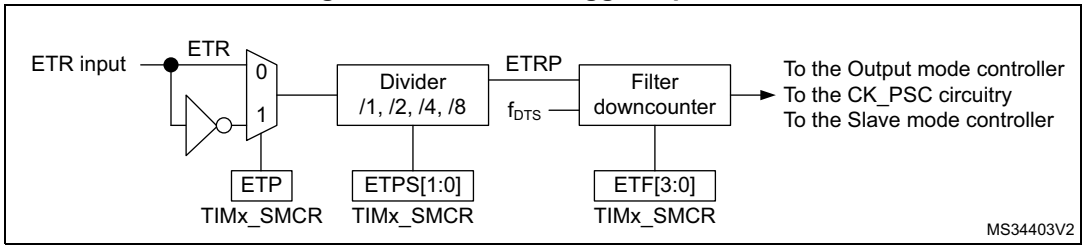
23.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see *Section 23.3.5*)
- trigger for the slave mode (see *Section 23.3.26*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 23.3.7*)

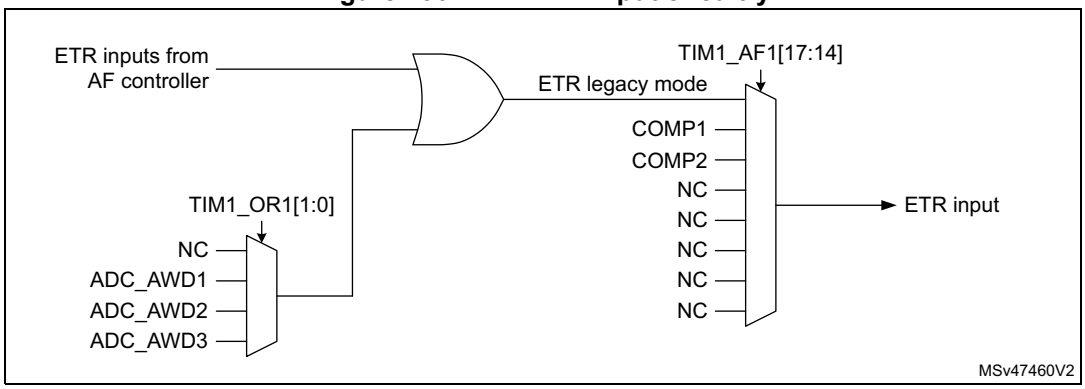
Figure 135 below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

Figure 135. External trigger input block



The ETR input comes from multiple sources: input pins (default configuration), comparator outputs and analog watchdogs. The selection is done with the ETRSEL[3:0] and the TIM1_OR1[1:0] bitfields.

Figure 136. TIM1 ETR input circuitry



23.3.5 Clock selection

The counter clock can be provided by the following clock sources:

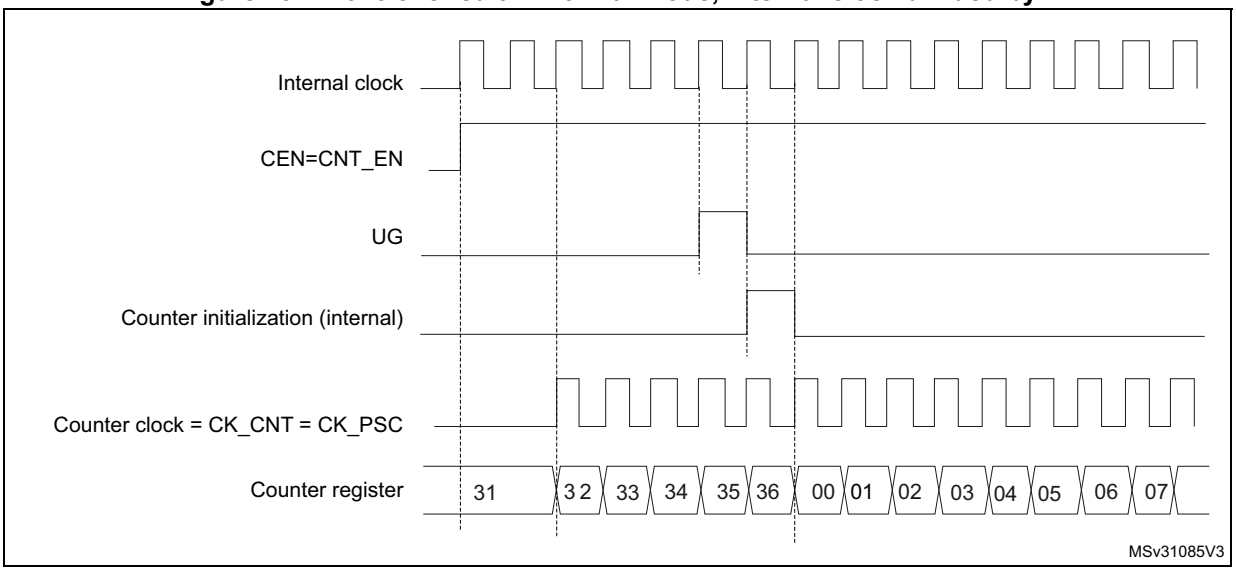
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 137 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

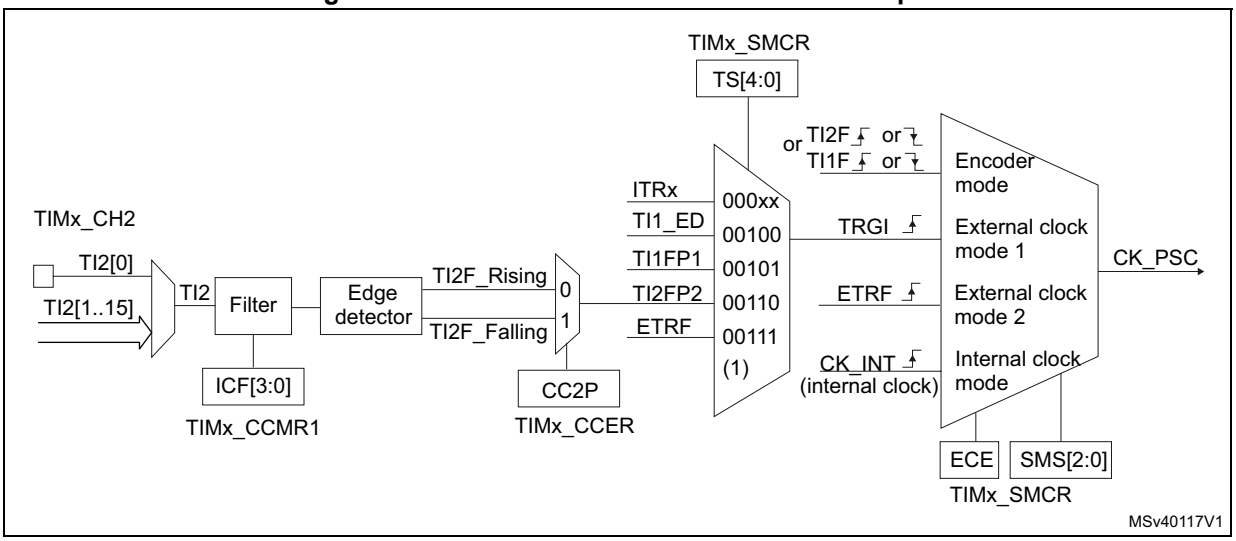
Figure 137. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 138. TI2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

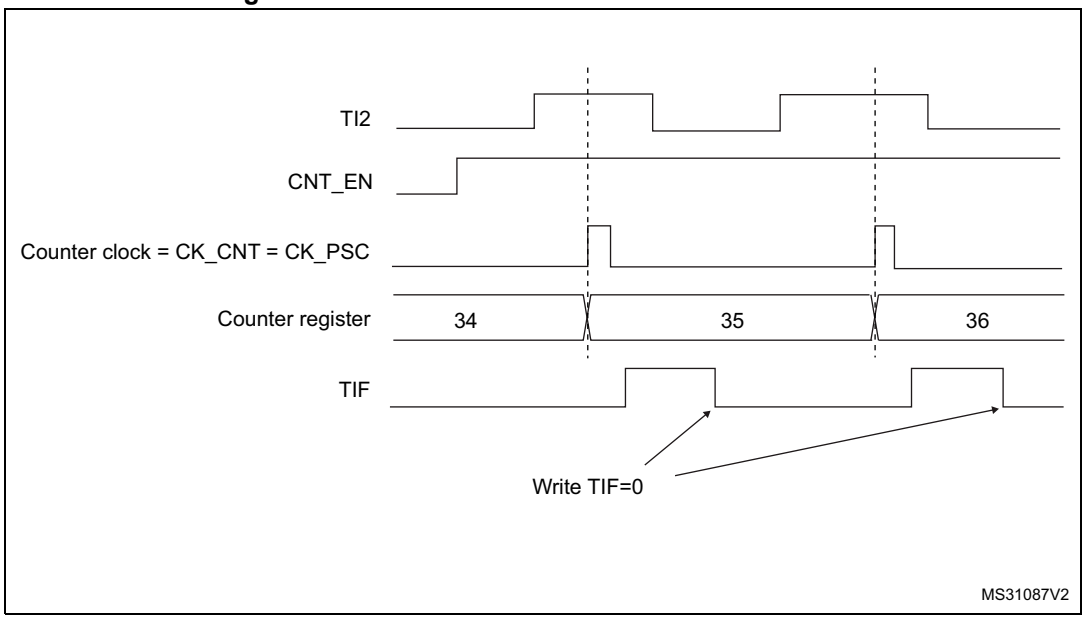
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so the user does not need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 139. Control circuit in external clock mode 1



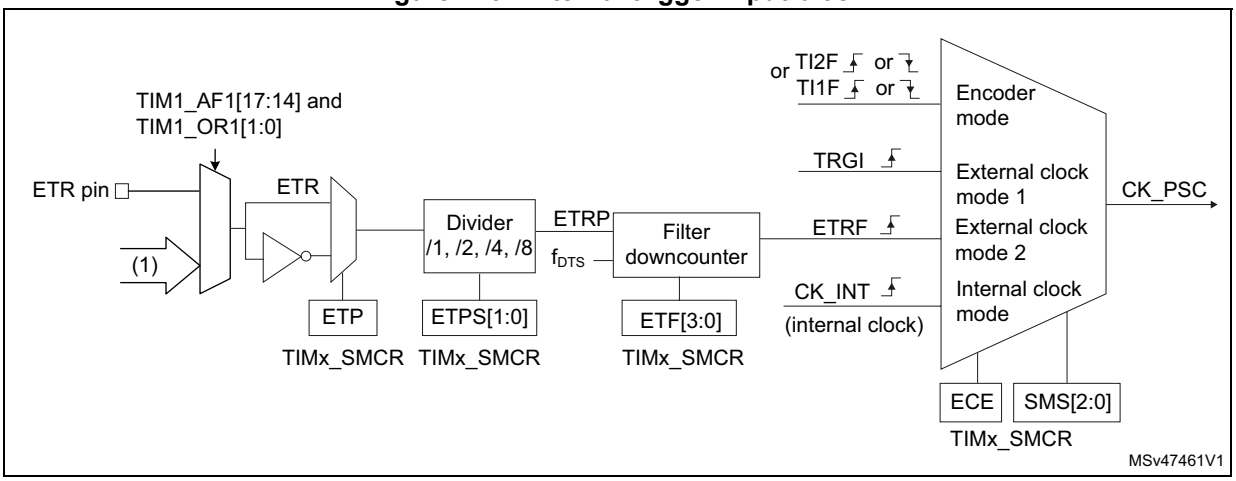
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The *Figure 140* gives an overview of the external trigger input block.

Figure 140. External trigger input block



1. Refer to *Figure 136: TIM1 ETR input circuitry*.

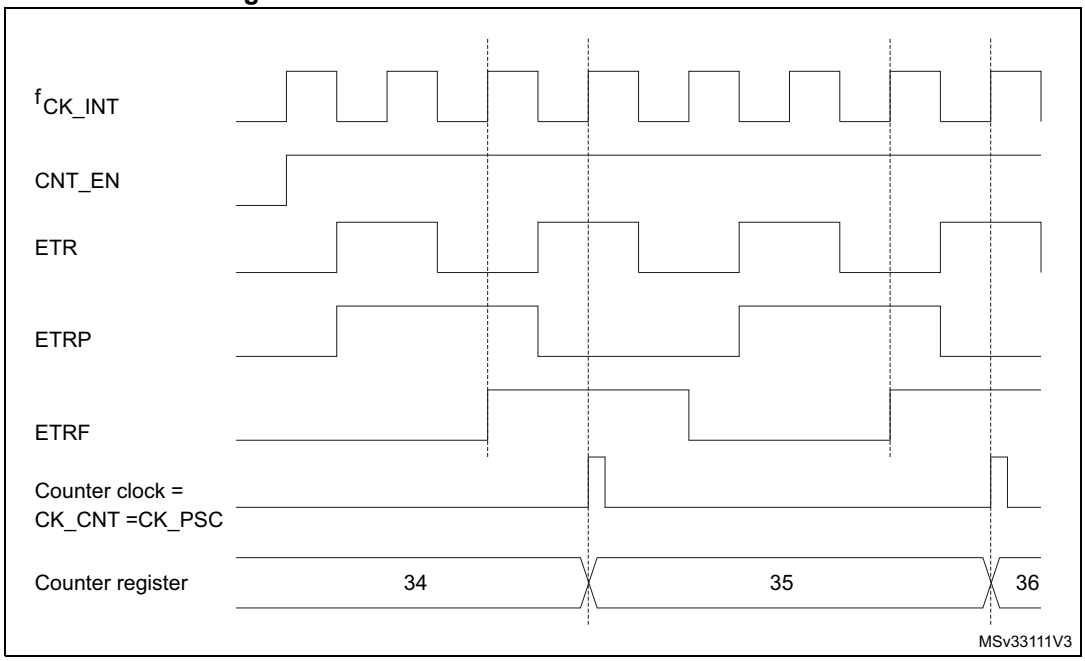
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by proper ETPS prescaler setting.

Figure 141. Control circuit in external clock mode 2



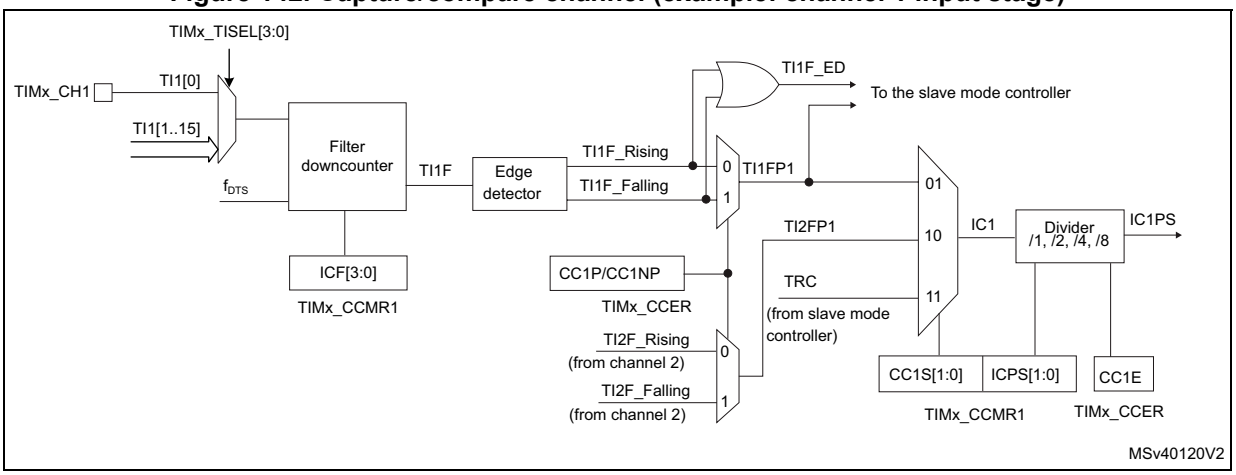
23.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 142 to *Figure 145* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 142. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 143. Capture/compare channel 1 main circuit

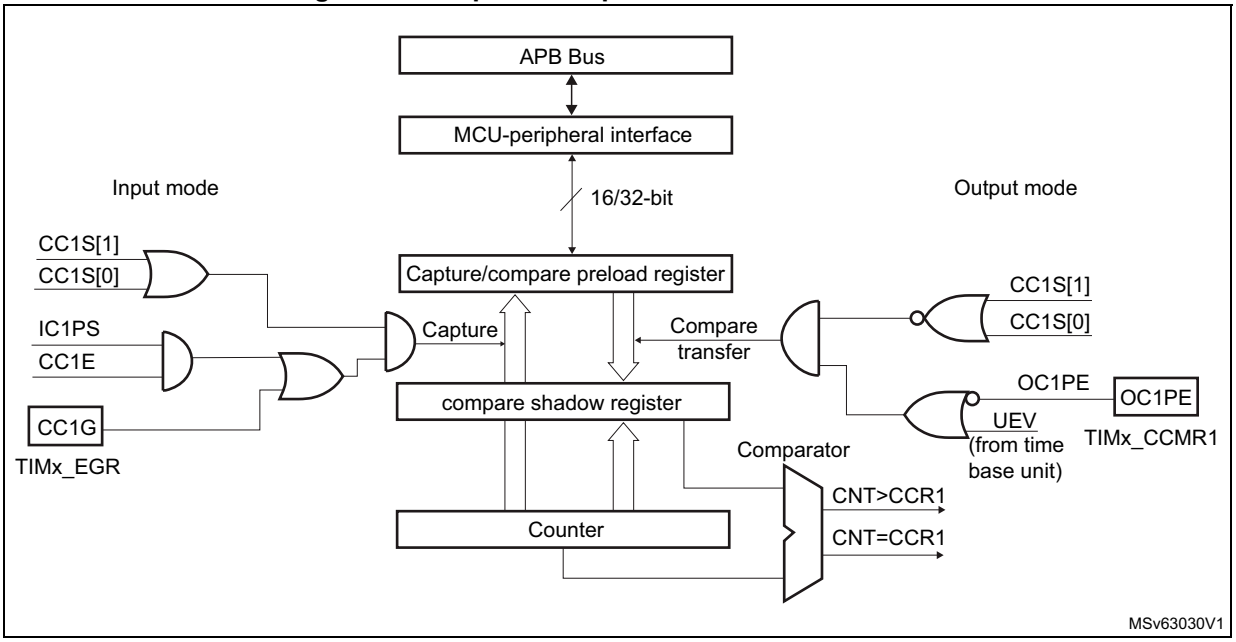
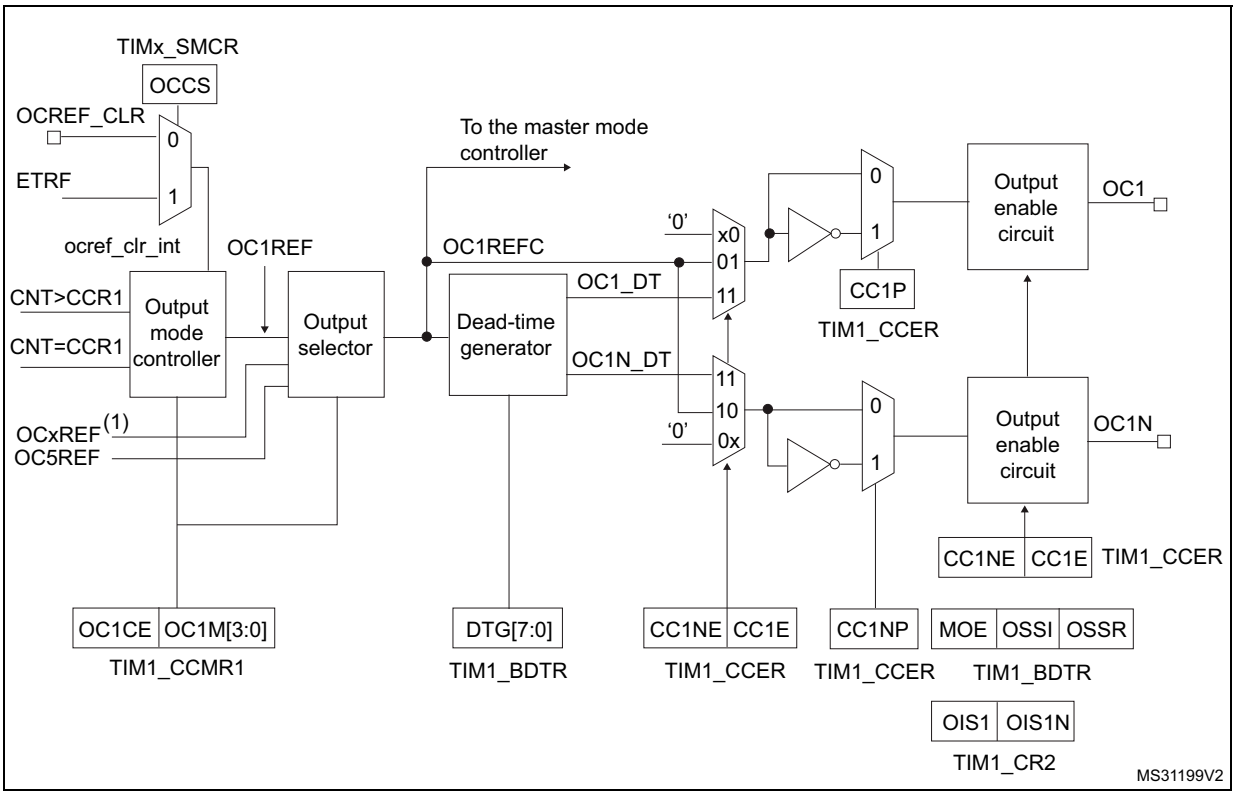


Figure 144. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)



1. OCxREF, where x is the rank of the complementary channel

Figure 145. Output stage of capture/compare channel (channel 4)

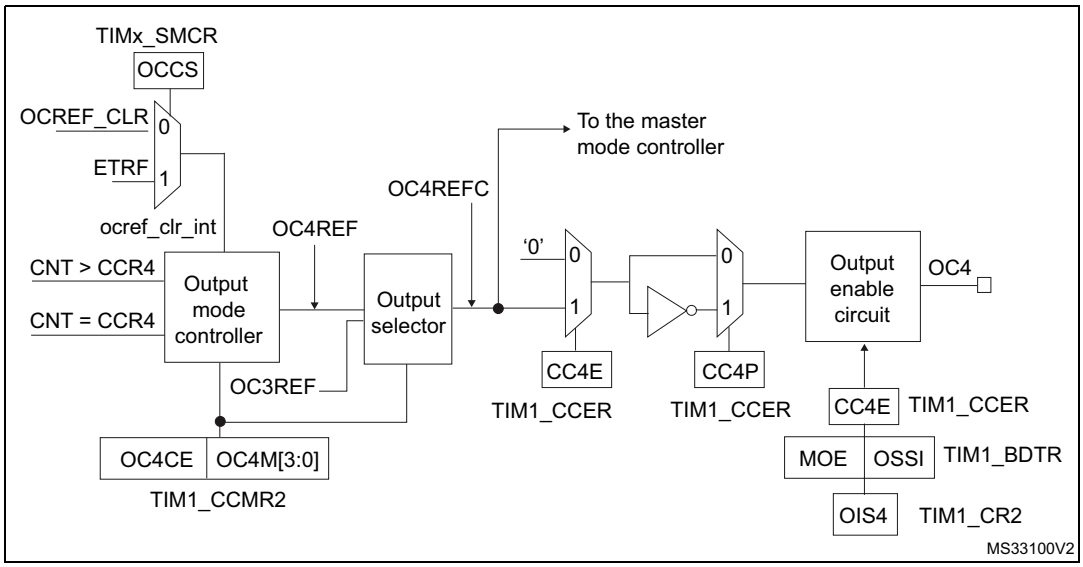
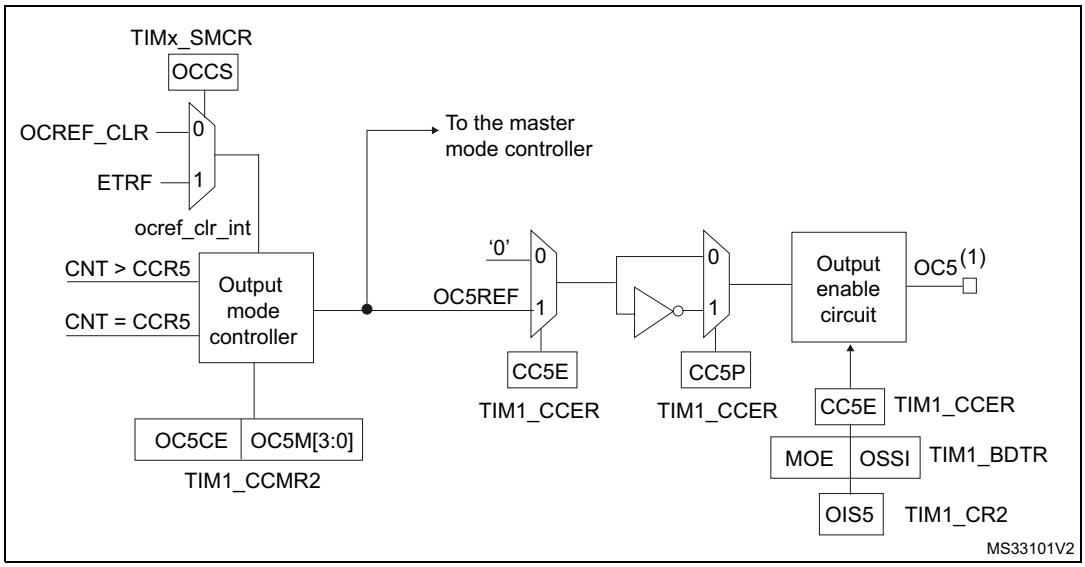


Figure 146. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

23.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written with '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

23.3.8 PWM input mode

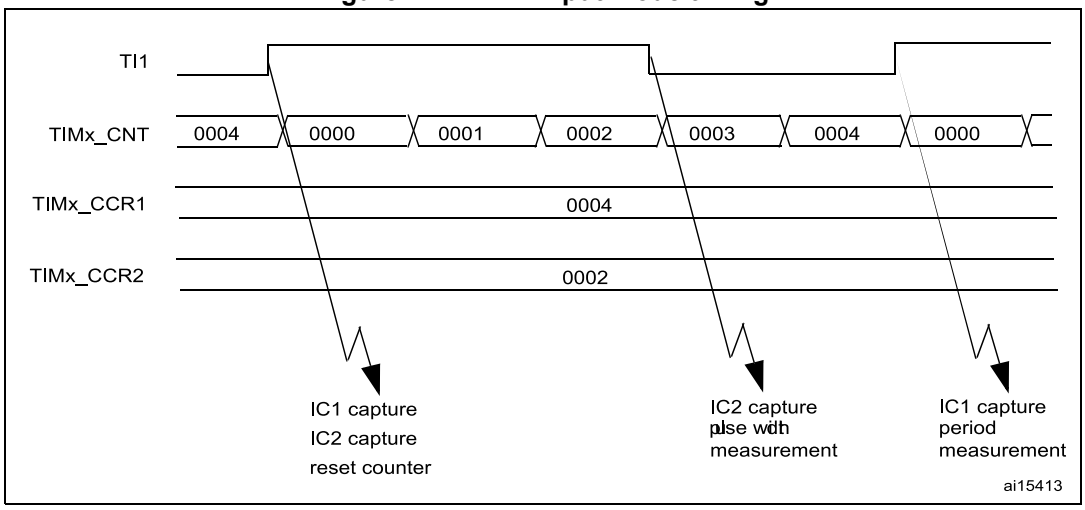
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, the user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 147. PWM input mode timing



23.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

23.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while Channel 5 and 6 are only available inside the device (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

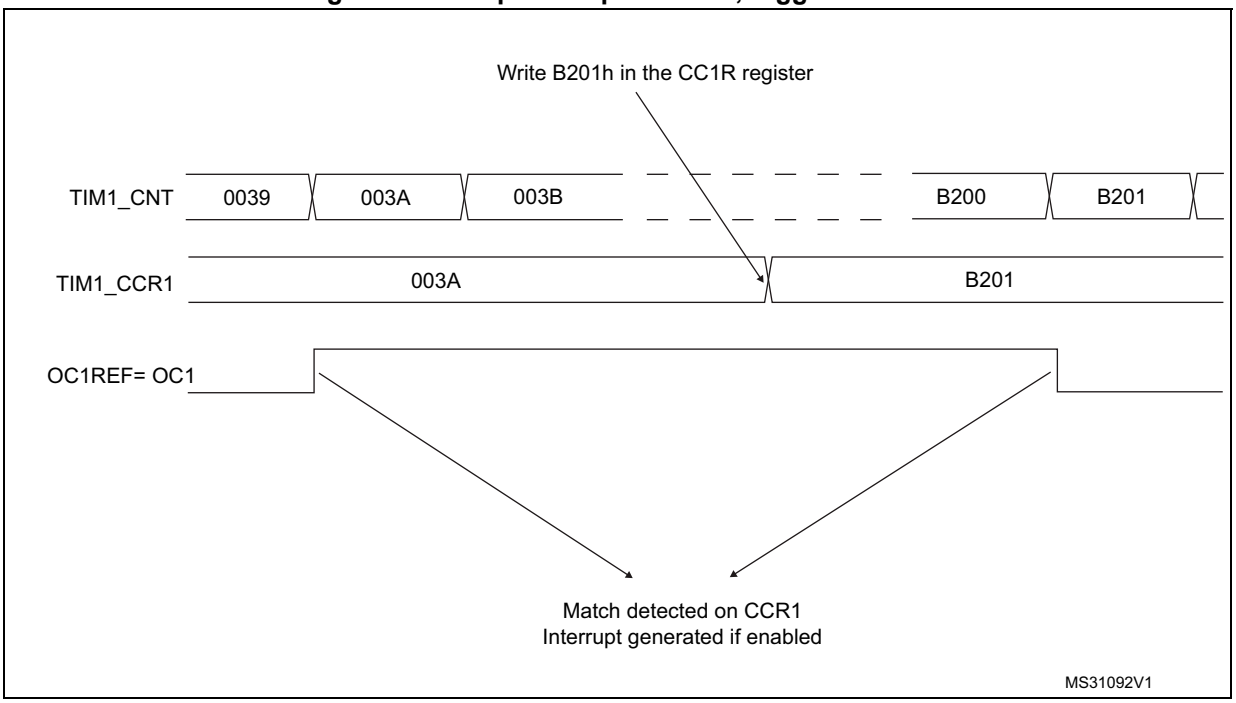
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 148*.

Figure 148. Output compare mode, toggle on OC1



23.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

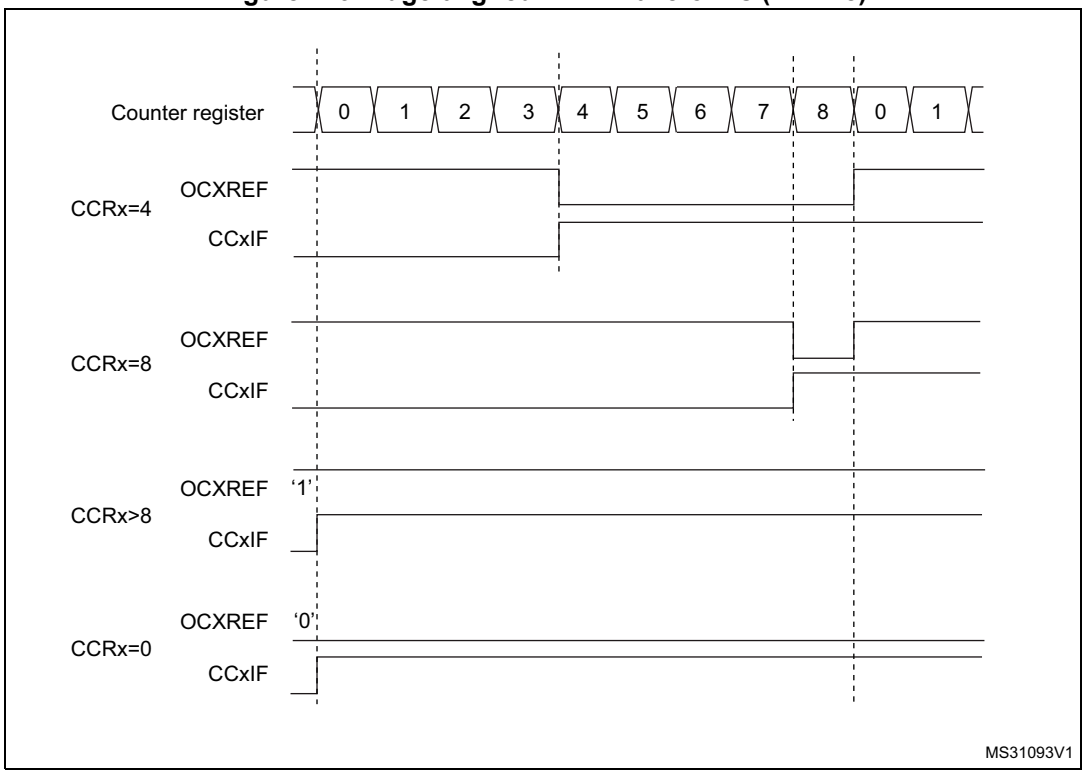
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 623*.

  In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. *Figure 149* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 149. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode on page 627*

  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

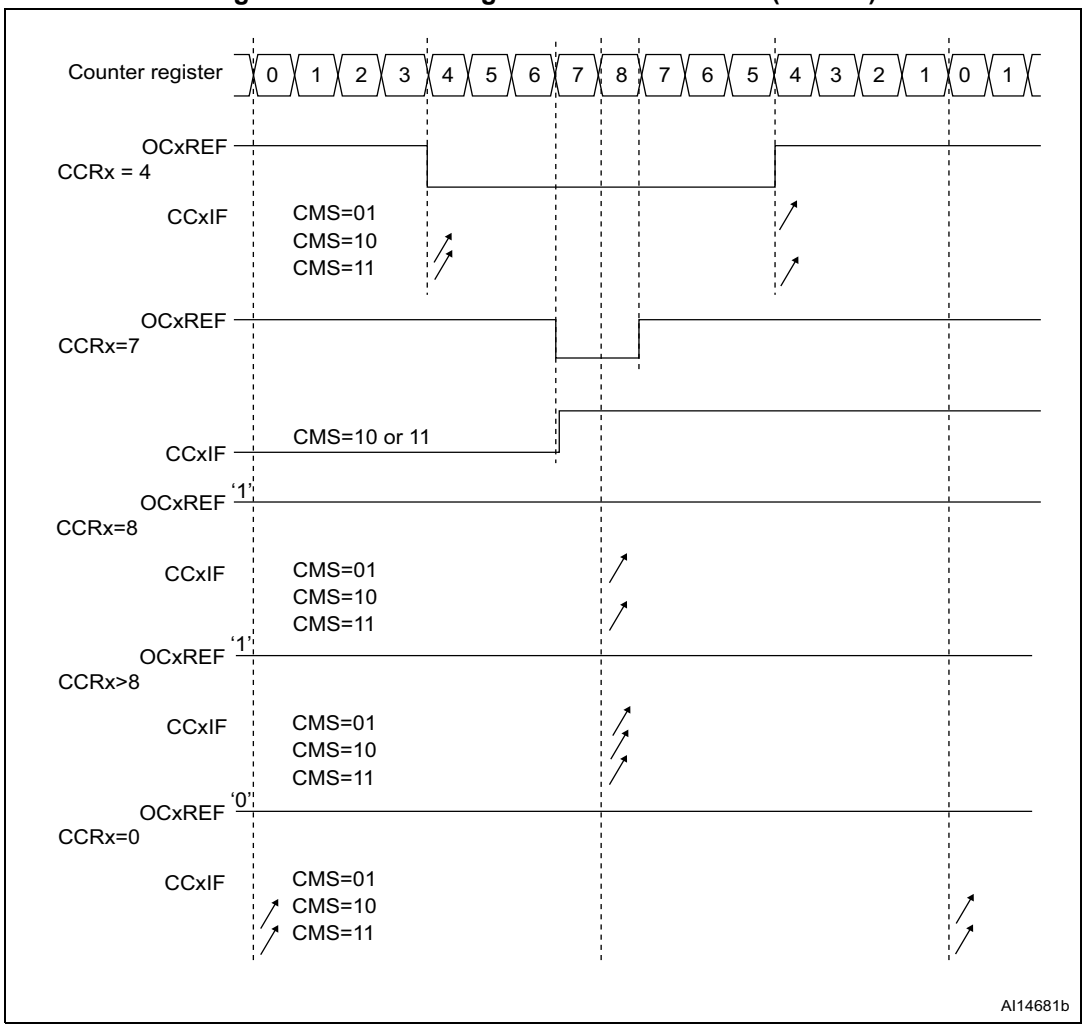
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the *Center-aligned mode (up/down counting) on page 630*.

Figure 150 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 150. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

23.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

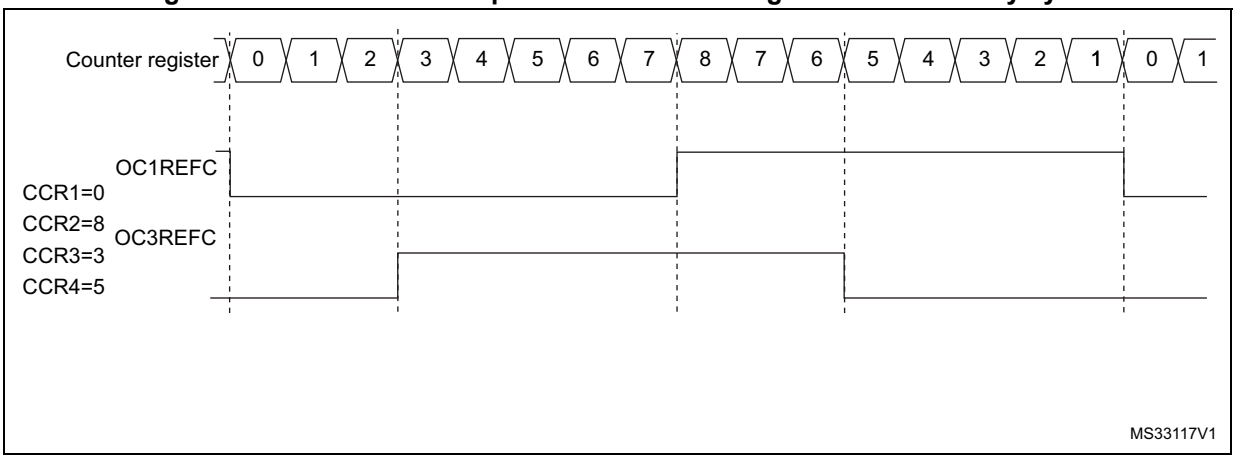
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

Figure 151 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 151. Generation of 2 phase-shifted PWM signals with 50% duty cycle



23.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

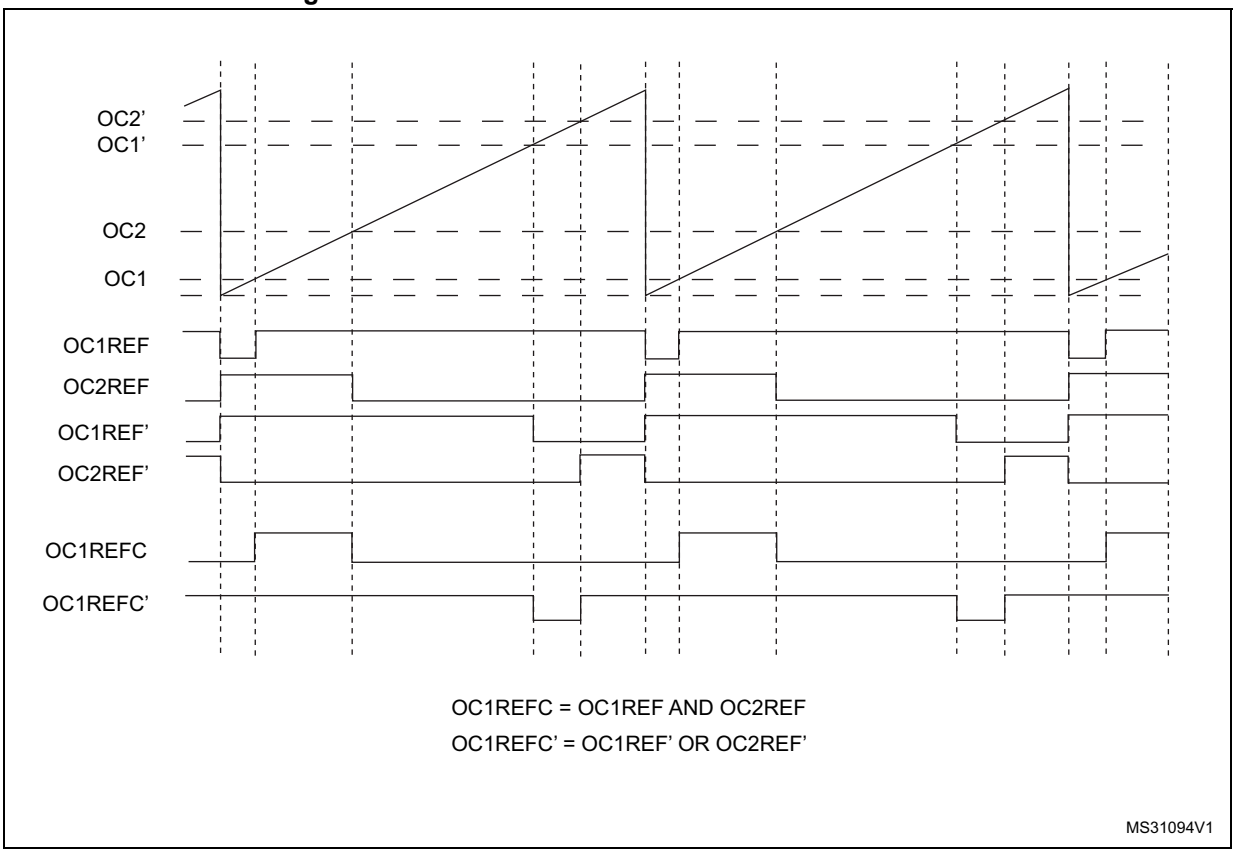
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 152 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.

Figure 152. Combined PWM mode on channel 1 and 3



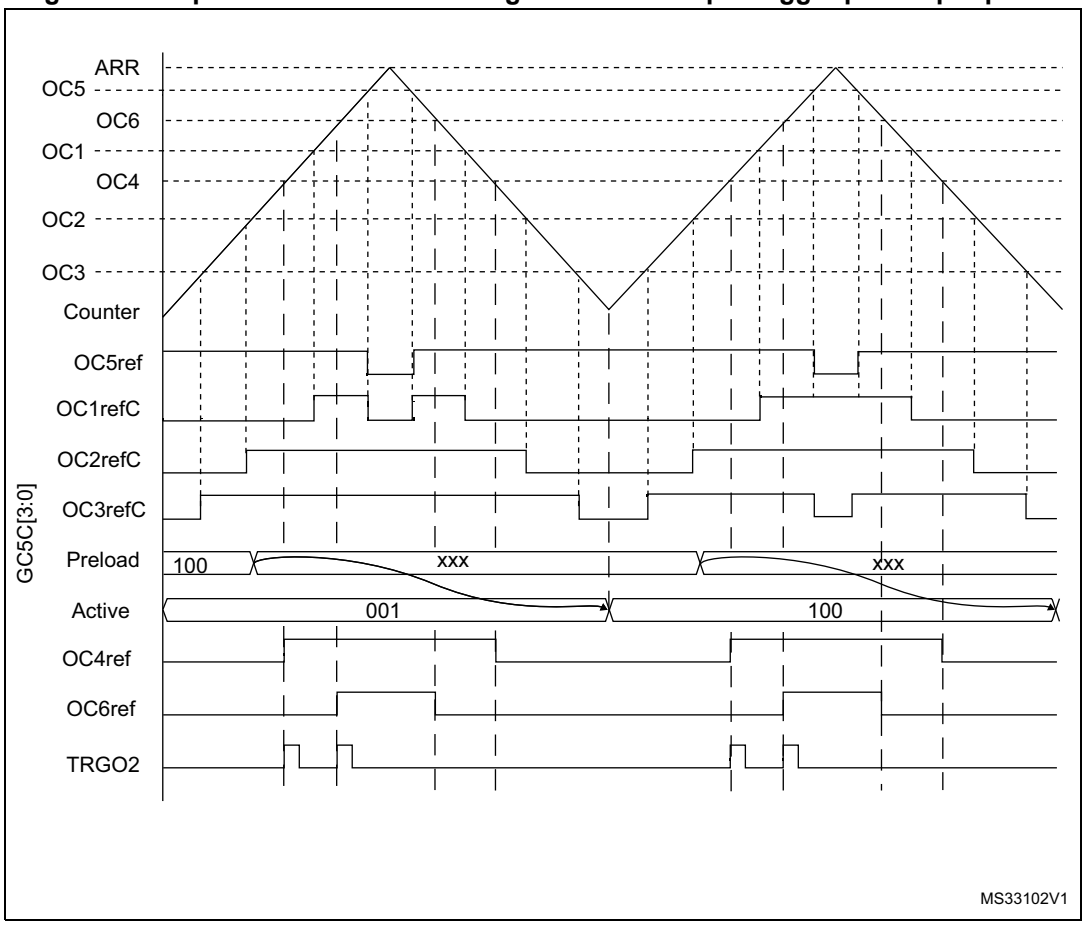
23.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, OC1REFC is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, OC2REFC is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, OC3REFC is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 153. 3-phase combined PWM signals with multiple trigger pulses per period



The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 23.3.27: ADC synchronization* for more details.

23.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 168: Output control bits for complementary OCx and OCxN channels with break feature on page 699* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 154. Complementary output with dead-time insertion

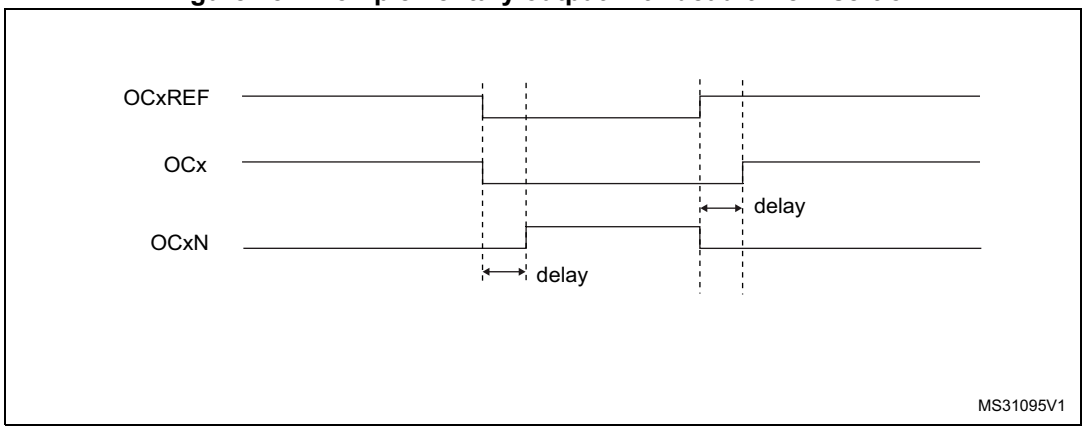


Figure 155. Dead-time waveforms with delay greater than the negative pulse

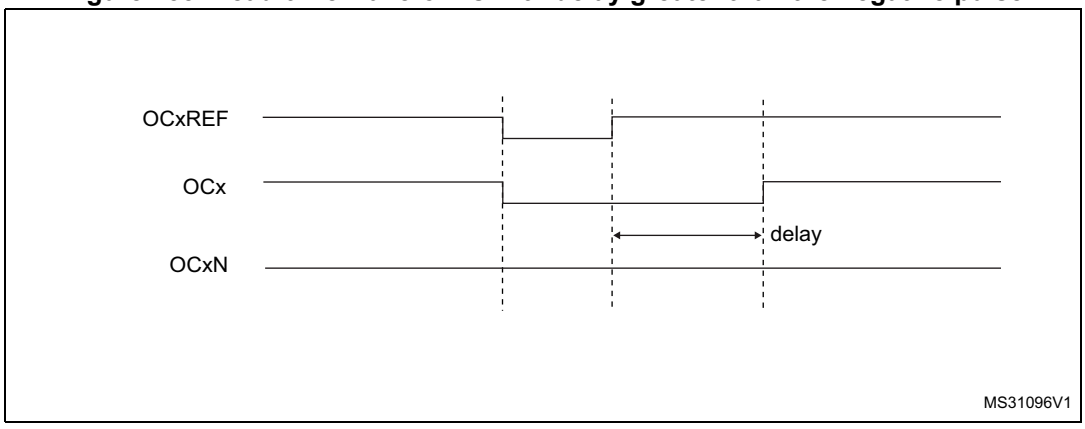
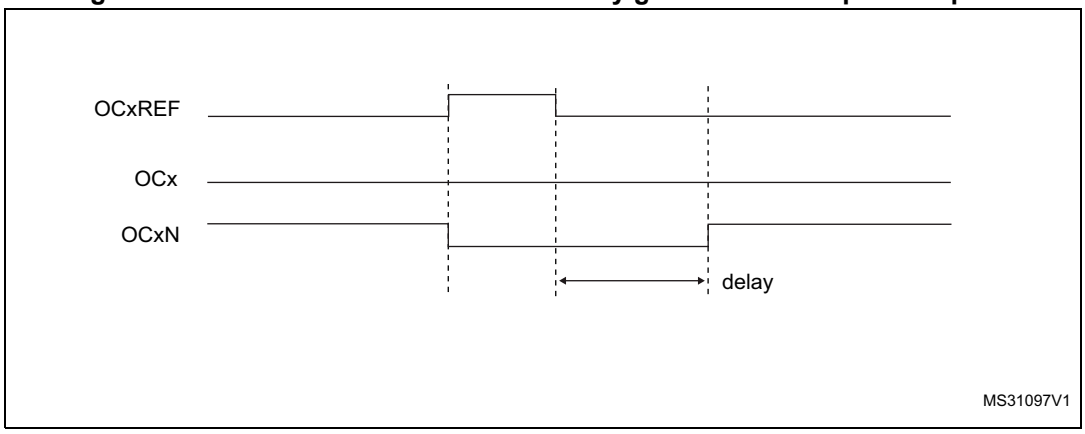


Figure 156. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 23.4.20: TIM1 break and dead-time register (TIM1_BDTR)* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

23.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 timer. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows the outputs to be enabled/disabled by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 168: Output control bits for complementary OCx and OCxN channels with break feature on page 699* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKE/BK2E and BKP/BK2P can be modified at the same time. When the BKE/BK2E and BKP/BK2P bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 and TIMx_AF2 registers.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the SYSCFG_CFGR2 register), with polarity selection and optional digital filtering
- An internal source:
  - the Cortex®-M4 LOCKUP output
  - the PVD output
  - the SRAM parity error signal
  - a Flash memory ECC double error detection
  - a clock failure event generated by the CSS detector
  - the output from a comparator, with polarity selection and optional digital filtering

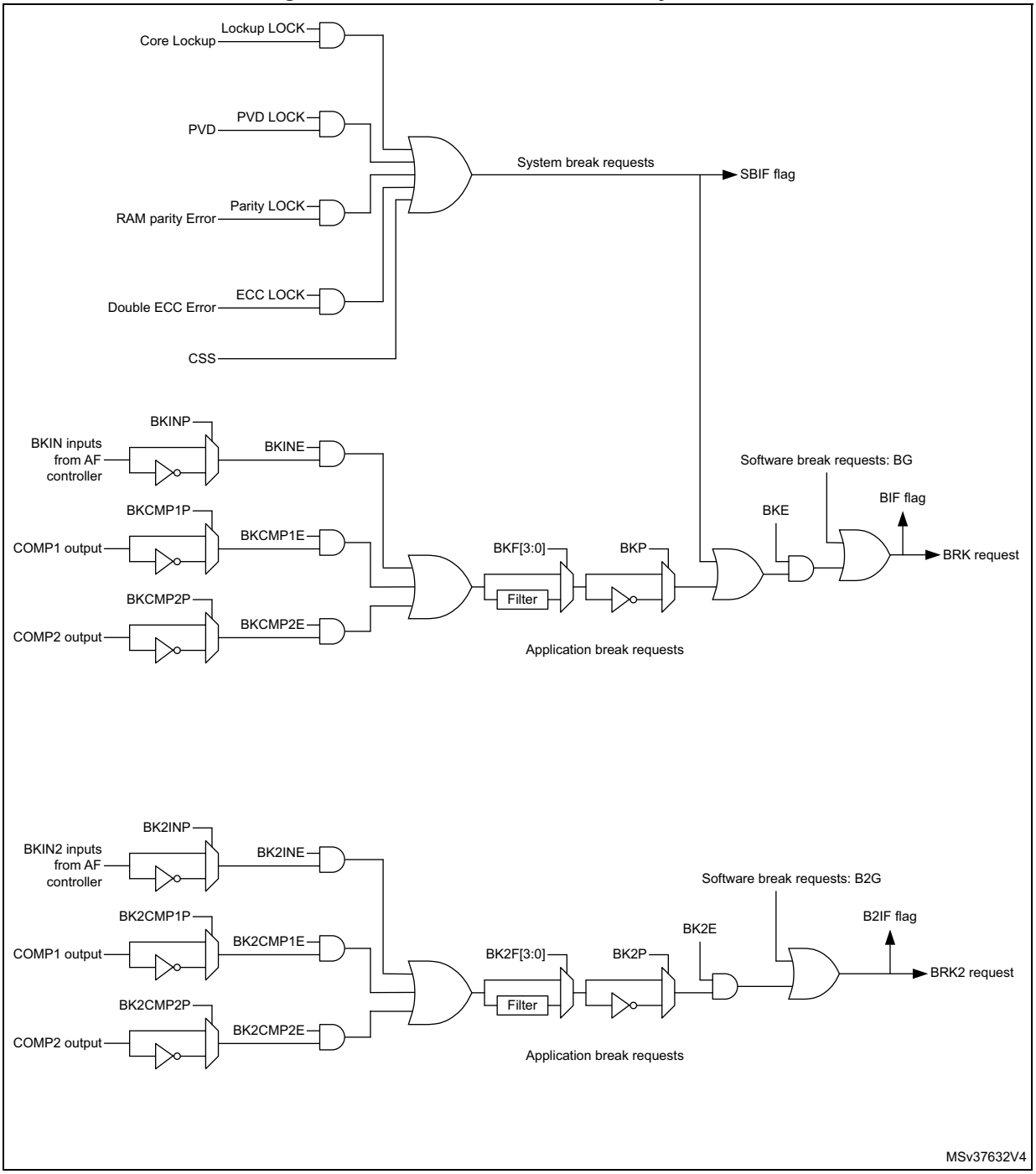
The sources for break2 (BRK2) are:

- An external source connected to one of the BKIN pin (as per selection done in the SYSCFG_CFGR2 register), with polarity selection and optional digital filtering
- An internal source coming from a comparator output.

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.

All sources are ORed before entering the timer BRK or BRK2 inputs, as per *Figure 157* below.

Figure 157. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

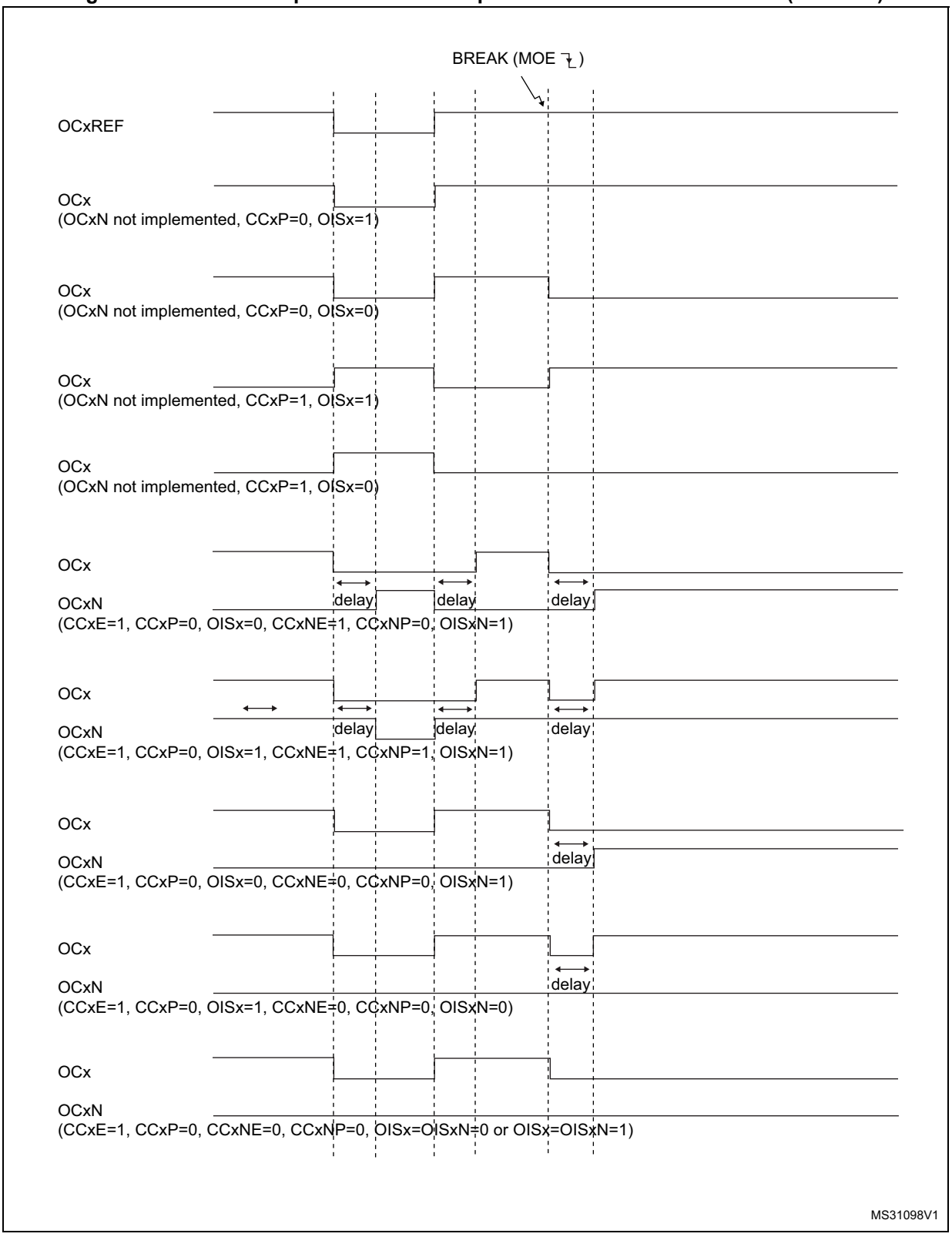
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs will be in idle state and forced to inactive level or Hi-Z depending on OSSI value.
If both the MOE and AOE bits are reset by the CPU, the outputs will be in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

Note: The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 23.4.20: TIM1 break and dead-time register (TIM1_BDTR)*. The LOCK bits can be written only once after an MCU reset.

Figure 158 shows an example of behavior of the outputs in response to a break.

Figure 158. Various output behavior in response to a break event on BRK (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in *Table 164*.

Note: BRK2 must only be used with OSSR = OSSI = 1.

Table 164. Behavior of timer outputs versus BRK/BRK2 inputs

BRK	BRK2	Timer outputs state	Typical use case	
			OCxN output (low side switches)	OCx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 159 gives an example of OCx and OCxN output behavior in case of active signals on BRK and BRK2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 159. PWM output state following BRK and BRK2 pins assertion (OSSI=1)

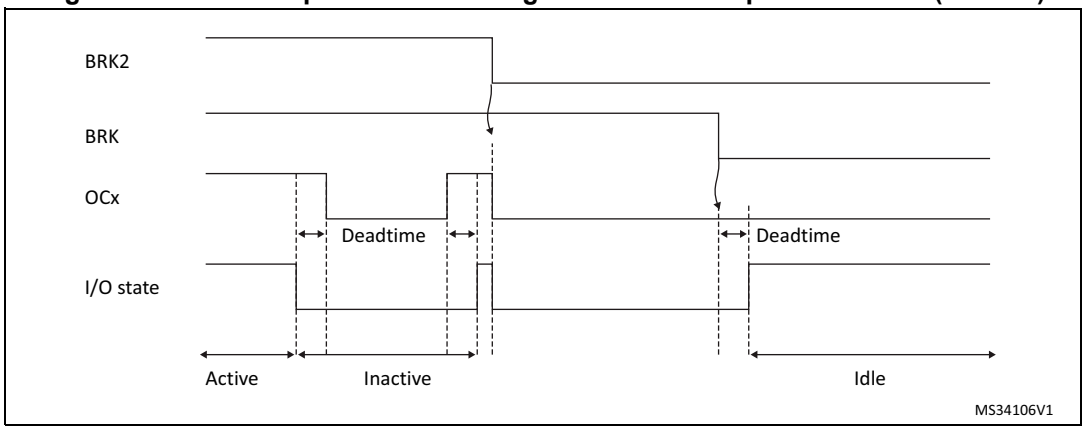
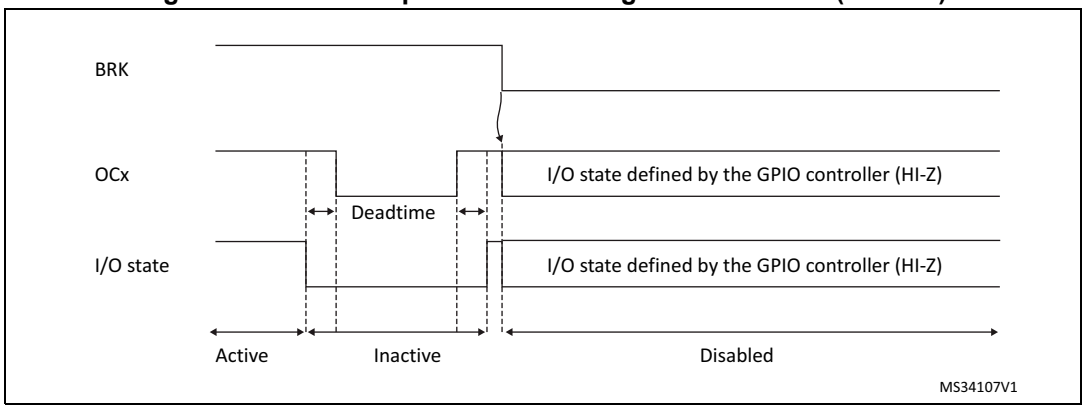


Figure 160. PWM output state following BRK assertion (OSSI=0)



23.3.17 Bidirectional break inputs

The TIM1 are featuring bidirectional break I/Os, as represented on *Figure 161*.

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break and break2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the break and break2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BK(2)E = 1). When a software break event is generated with BK(2)E = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break(2) I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BK(2)DSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 165*)

Table 165. Break protection disarming conditions

MOE	BKDIR (BK2DIR)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

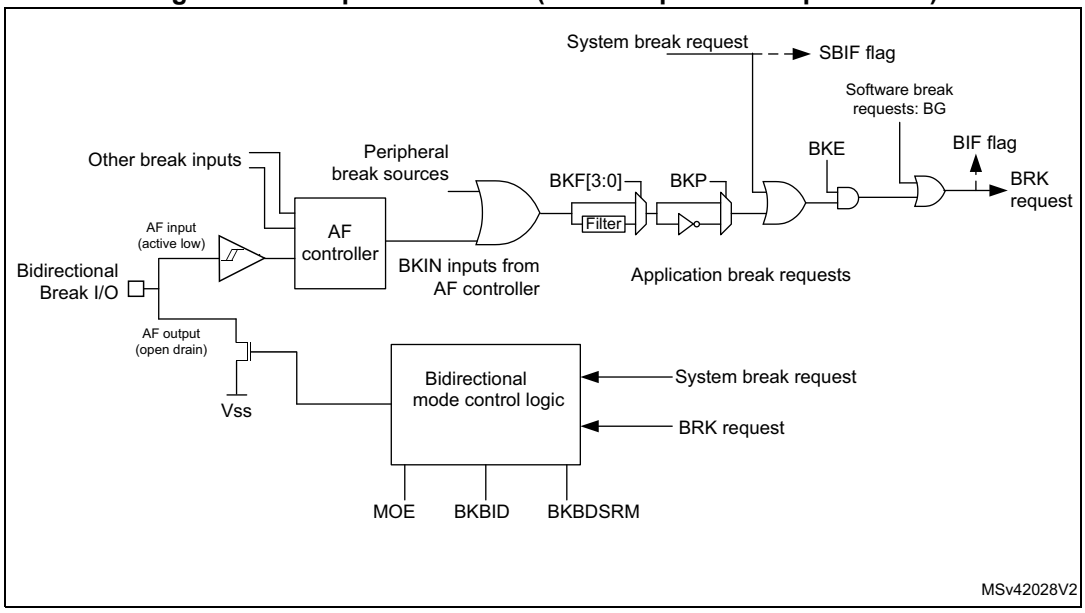
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 161. Output redirection (BRK2 request not represented)



23.3.18 Clearing the OCxREF signal on an external event

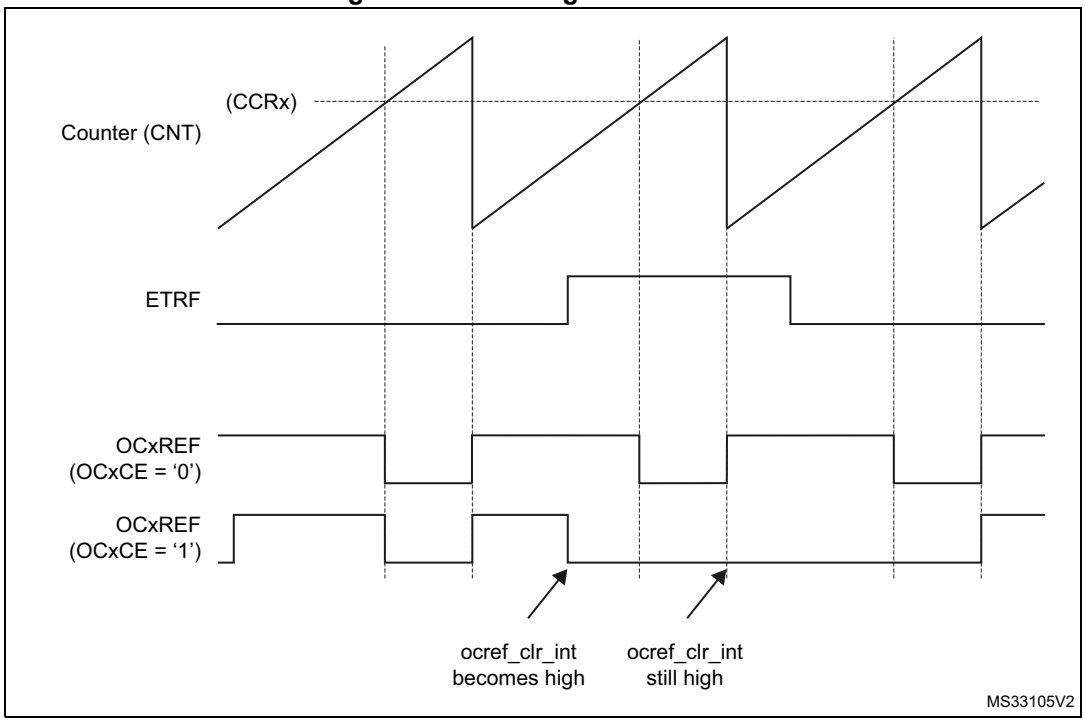
The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. ocref_clr_int input can be selected between the OCREF_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

When ETRF is chosen, ETR must be configured as follows:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

Figure 162 shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 162. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.

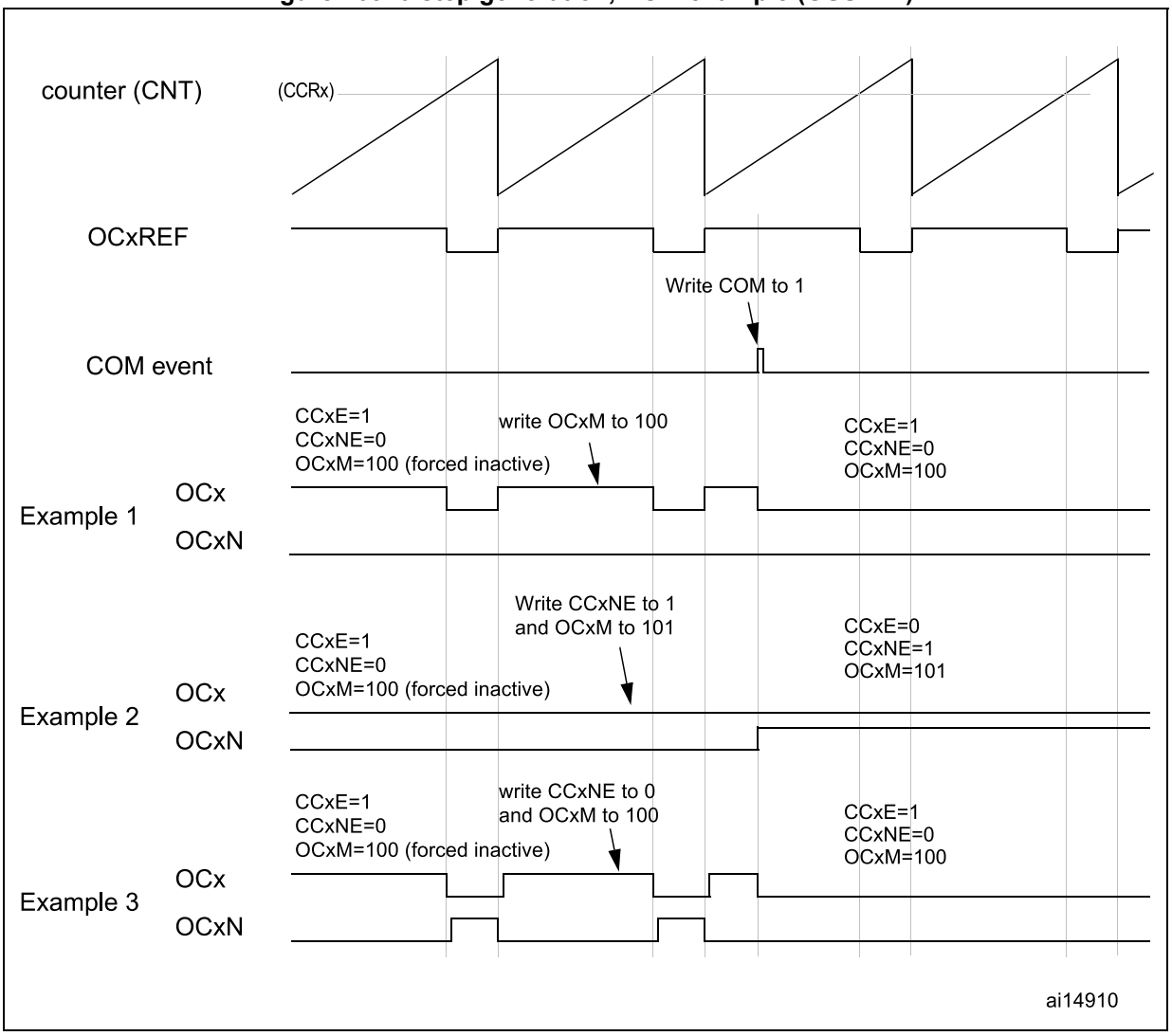
23.3.19 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 163* describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 163. 6-step generation, COM example (OSSR=1)



23.3.20 One-pulse mode

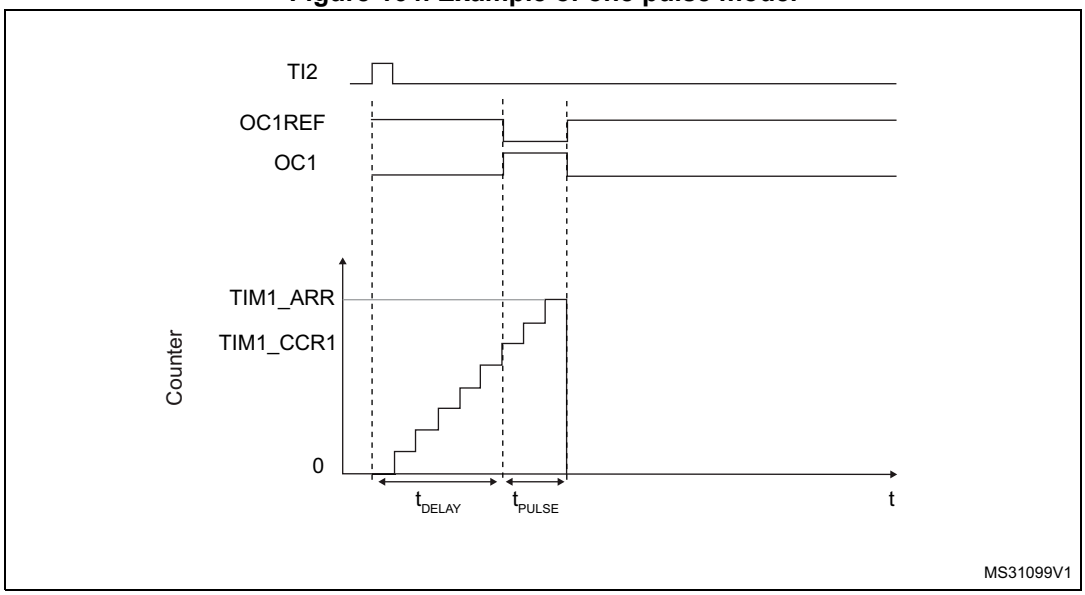
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 164. Example of one pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

23.3.21 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 23.3.20*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

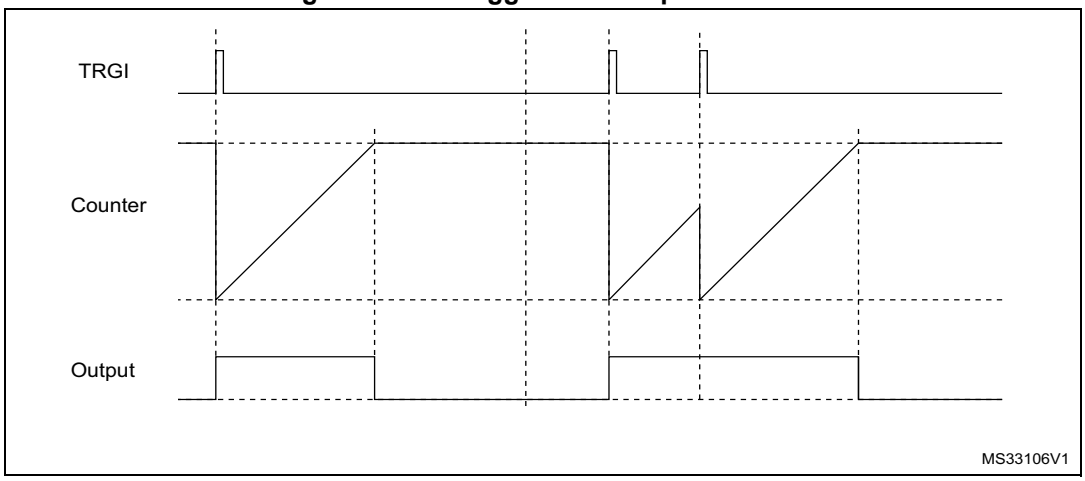
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 165. Retriggerable one pulse mode



23.3.22 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to *Table 166*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Note: The prescaler must be set to zero when encoder mode is enabled

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 166. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The *Figure 166* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 166. Example of counter operation in encoder interface mode.

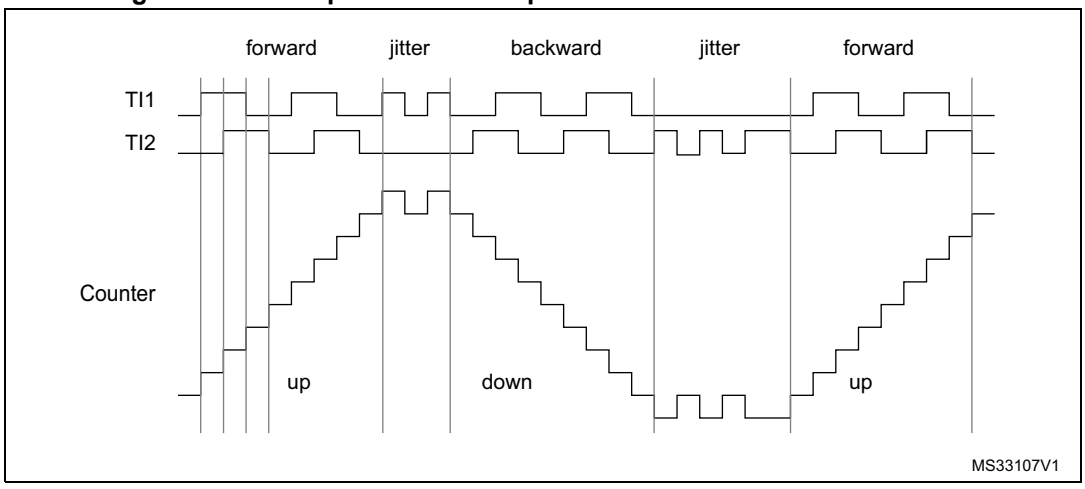
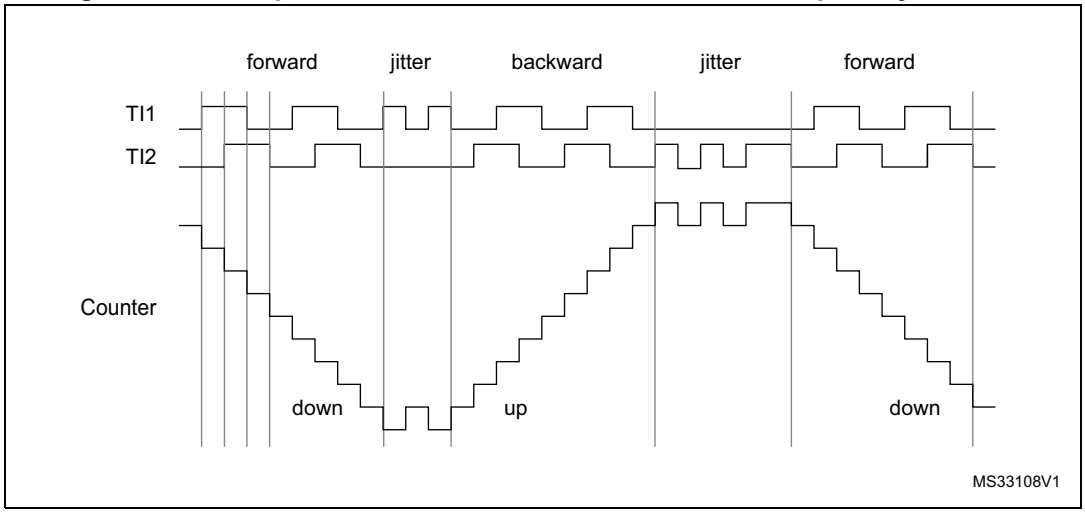


Figure 167 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 167. Example of encoder interface mode with TI1FP1 polarity inverted.



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

23.3.23 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

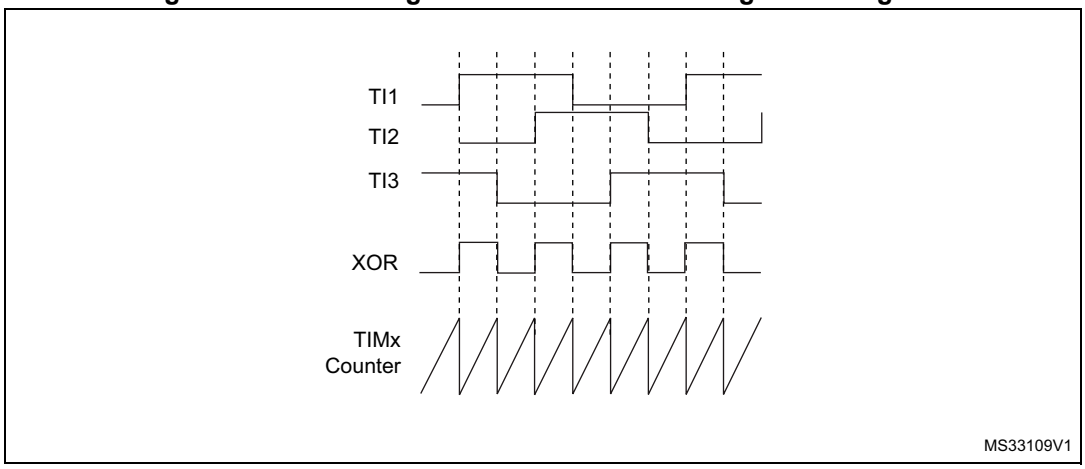
There is no latency between the UIF and UIFCPY flags assertion.

23.3.24 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 168* below.

Figure 168. Measuring time interval between edges on 3 signals



23.3.25 Interfacing with Hall sensors

This is done using the advanced-control timer (TIM1) to generate PWM signals to drive the motor and another timer TIMx (TIM2) referred to as “interfacing timer” in *Figure 169*. The “interfacing timer” captures the 3 timer input pins (CC1, CC2, CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (See *Figure 142: Capture/compare channel (example: channel 1 input stage) on page 641*). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1) through the TRGO output.

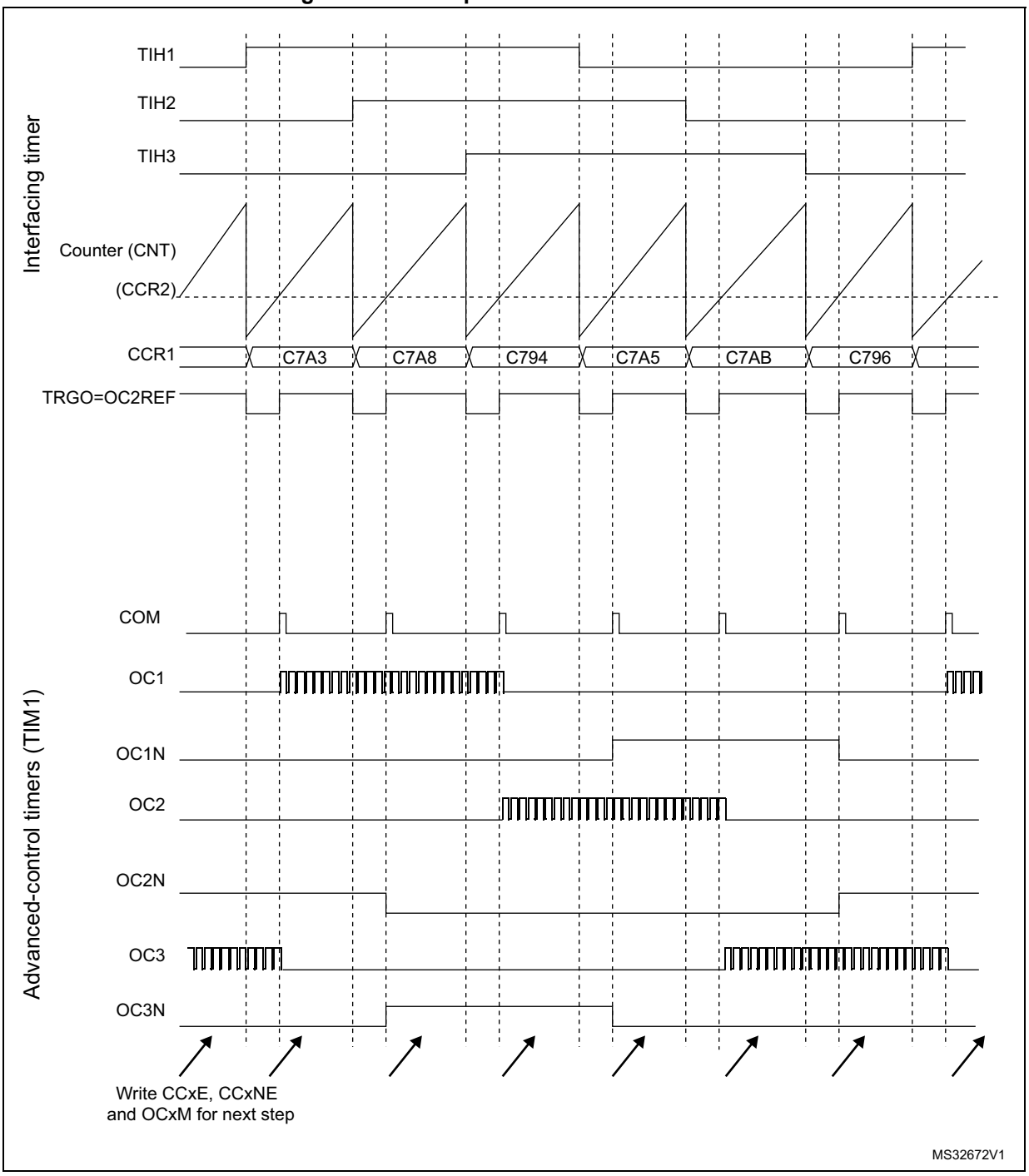
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘01’. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The *Figure 169* describes this example.

Figure 169. Example of Hall sensor interface



23.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 24.3.19: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

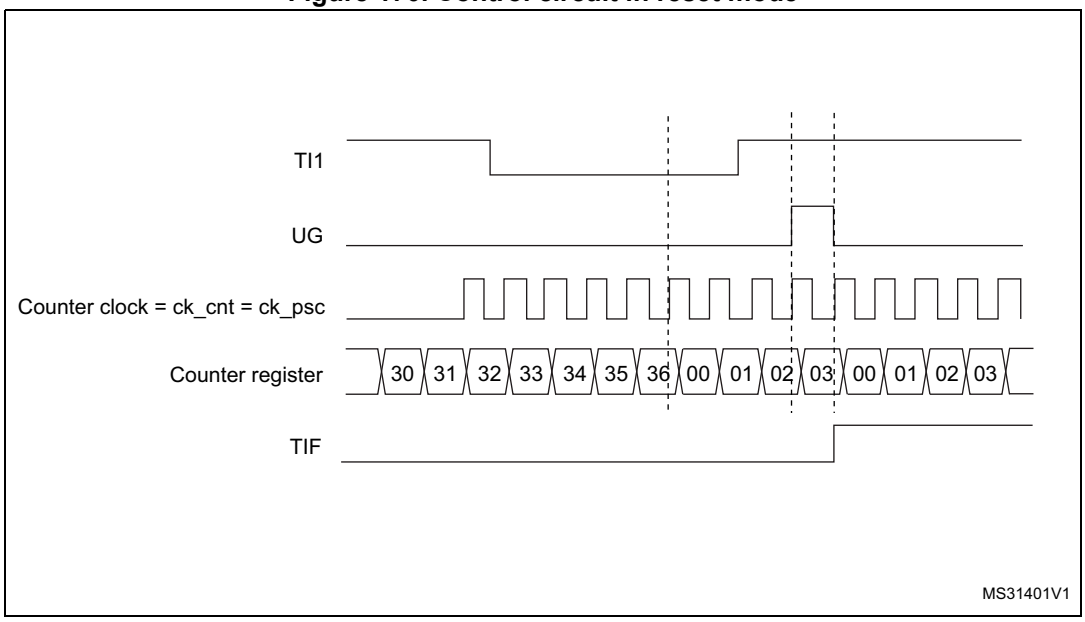
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 170. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

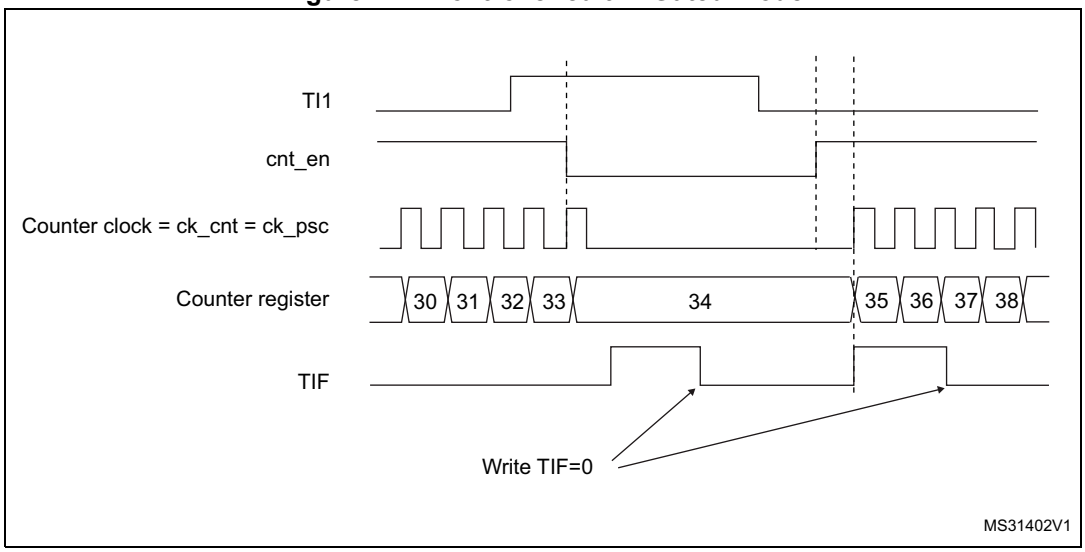
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 171. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1

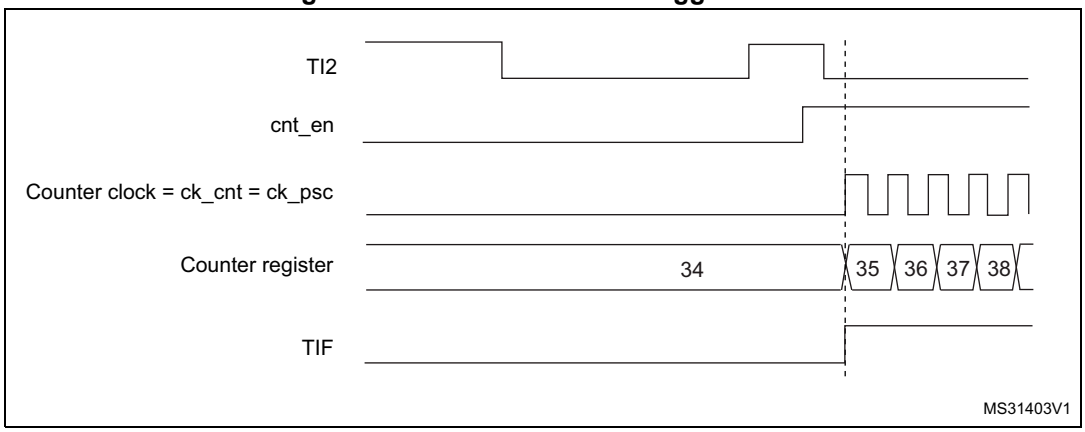
register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 172. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

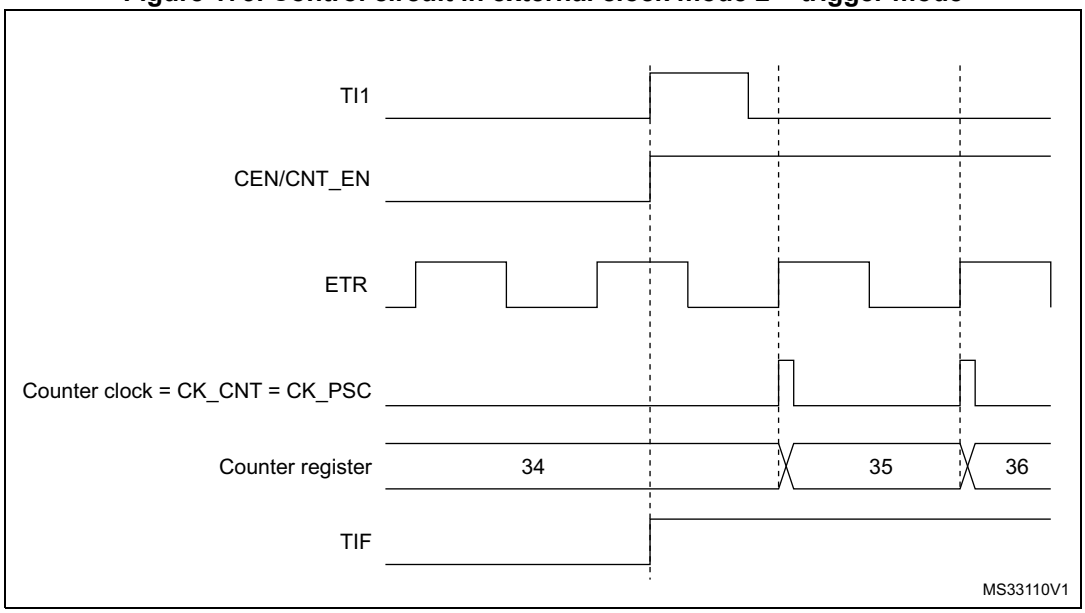
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 173. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

23.3.27 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 153 on page 653*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

23.3.28 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

23.3.29 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIM1_STOP configuration bit in DBGMCU module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

23.4 TIM1 registers

Refer to for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

23.4.1 TIM1 control register 1 (TIM1_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx):
00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Note: $t_{DTS} = 1/f_{DTS}$, $t_{CK\_INT} = 1/f_{CK\_INT}$.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: Switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) is not allowed

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

23.4.2 TIM1 control register 2 (TIM1_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
								rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).

0100: **Compare** - OC1REFC signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REFC signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REFC signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REFC signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REFC signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REFC signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REFC rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REFC rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REFC or OC6REFC rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REFC rising or OC6REFC falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REFC or OC6REFC rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REFC rising or OC6REFC falling edges generate pulses on TRGO2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output Idle state 6 (OC6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output Idle state 5 (OC5 output)
Refer to OIS1 bit

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)
0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)
0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow selected information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)
Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.

23.4.3 TIM1 slave mode control register (TIM1_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity
This bit selects whether ETR or $\overline{\text{ETR}}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.
Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of f_{CK_INT} frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010: Internal Trigger 2 (ITR2)
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
00110: Filtered Timer Input 2 (TI2FP2)
00111: External Trigger input (ETRF)
Others: Reserved

See *Table 167: TIM1 internal trigger connection on page 685* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: OCREF_CLR_INT is connected to the OCREF_CLR input
1: OCREF_CLR_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Codes above 1000: Reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=00100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 167. TIM1 internal trigger connection

Slave TIM	ITR0 (TS = 00000)	ITR1 (TS = 00001)	ITR2 (TS = 00010)	ITR3 (TS = 00011)
TIM1	-	TIM2	-	TIM17 OC1

23.4.4 TIM1 DMA/interrupt enable register (TIM1_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

23.4.5 TIM1 status register (TIM1_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF
														rc_w0	rc_w0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description (Note: Channel 6 can only be configured as output)

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description (Note: Channel 5 can only be configured as output)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System Break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 23.4.3: TIM1 slave mode control register (TIM1_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

23.4.6 TIM1 event generation register (TIM1_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows CCxE, CCxNE and OCxM bits to be updated.

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. The prescaler internal counter is also cleared (the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

23.4.7 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter
Refer to IC1F[3:0] description.

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler
Refer to IC1PSC[1:0] description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

23.4.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output Compare 2 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode
Refer to OC1M[3:0] description.

Bit 11 **OC2PE**: Output Compare 2 preload enable
Refer to OC1PE description.

Bit 10 **OC2FE**: Output Compare 2 fast enable
Refer to OC1FE description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE:** Output Compare 1 clear enable
0: OC1Ref is not affected by the ocref_clr_int signal
1: OC1Ref is cleared as soon as a High level is detected on ocref_clr_int signal (OCREF_CLR input or ETRF input)

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

23.4.9 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter
Refer to IC1F[3:0] description.

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
Refer to IC1PSC[1:0] description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter
Refer to IC1F[3:0] description.

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler
Refer to IC1PSC[1:0] description.

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

23.4.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Output compare mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] description.

Bit 11 **OC4PE**: Output compare 4 preload enable
Refer to OC1PE description.

Bit 10 **OC4FE**: Output compare 4 fast enable
Refer to OC1FE description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable
Refer to OC1CE description.

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M[3:0] description.

Bit 3 **OC3PE**: Output compare 3 preload enable
Refer to OC1PE description.

Bit 2 **OC3FE**: Output compare 3 fast enable
Refer to OC1FE description.

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

23.4.11 TIM1 capture/compare enable register (TIM1_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/Compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: OC1N active high.
1: OC1N active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/Compare 1 output polarity
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: The configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 168* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 168. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) OCx=0, OCxN=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN = OCxREF xor CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN = OCxREF x or CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Off-State (output enabled with inactive state) OCxN=CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered). Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration). **Note**: BRK2 can only be used if OSSI = OSSR = 1.	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

23.4.12 TIM1 counter (TIM1_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

23.4.13 TIM1 prescaler (TIM1_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

23.4.14 TIM1 auto-reload register (TIM1_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 23.3.1: Time-base unit on page 621* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

23.4.15 TIM1 repetition counter register (TIM1_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

the number of PWM periods in edge-aligned mode

the number of half PWM period in center-aligned mode.

23.4.16 TIM1 capture/compare register 1 (TIM1_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

23.4.17 TIM1 capture/compare register 2 (TIM1_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

23.4.18 TIM1 capture/compare register 3 (TIM1_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

23.4.19 TIM1 capture/compare register 4 (TIM1_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

23.4.20 TIM1 break and dead-time register (TIM1_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional

Refer to BKBID description

Bit 28 **BKBID**: Break Bidirectional

0: Break input BRK in input mode

1: Break input BRK in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 Disarm

Refer to BKDSRM description

Bit 26 **BKDSRM**: Break Disarm

0: Break input BRK is armed

1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity

0: Break input BRK2 is active low

1: Break input BRK2 is active high

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable

Note: The must only be used with OSSR = OSSI = 1.

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK2 acts asynchronously
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).

See OC/OCN enable description for more details (*Section 23.4.11: TIM1 capture/compare enable register (TIM1_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 157: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 23.4.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 23.4.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and B*K2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0]* bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0] x t_{DTG} with $t_{DTG} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x t_{DTG} with $t_{DTG} = 2 \times t_{DTS}$.
DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t_{DTG} with $t_{DTG} = 8 \times t_{DTS}$.
DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t_{DTG} with $t_{DTG} = 16 \times t_{DTS}$.
Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

23.4.21 TIM1 DMA control register (TIM1_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
10001: 18 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIMx_CR1.

– If DBL = 7 bytes and DBA = TIMx_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

– If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.
– If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

23.4.22 TIM1 DMA address for full transfer (TIM1_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

23.4.23 TIM1 option register 1 (TIM1_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_ RMP	Res.	Res.	TIM1_ETR_ ADC_RMP[1:0]	
											rw			rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TI1_RMP**: Input Capture 1 remap

0: TIM1 input capture 1 is connected to I/O

1: TIM1 input capture 1 is connected to COMP1 output

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **TIM1_ETR_ADC_RMP[1:0]**: TIM1_ETR_ADC remapping capability

00: TIM1_ETR is not connected to ADC AWDx (must be selected when the ETR comes from the ETR input pin)

01: TIM1_ETR is connected to ADC AWD1

10: TIM1_ETR is connected to ADC AWD2

11: TIM1_ETR is connected to ADC AWD3

Note: ADC AWDx sources are 'ORed' with the TIM1_ETR input signals. When ADC AWDx is used, it is necessary to make sure that the corresponding TIM1_ETR input pin is not enabled in the alternate function controller.

23.4.24 TIM1 capture/compare mode register 3 (TIM1_CCMR3)

Address offset: 0x54

Reset value: 0x0000 0000

The channels 5 and 6 can only be configured in output.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable
Refer to OC1CE description.

Bits 24, 14, 13, 12 **OC6M[3:0]**: Output compare 6 mode
Refer to OC1M description.

Bit 11 **OC6PE**: Output compare 6 preload enable
Refer to OC1PE description.

Bit 10 **OC6FE**: Output compare 6 fast enable
Refer to OC1FE description.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable
Refer to OC1CE description.

Bits 16, 6, 5, 4 **OC5M[3:0]**: Output compare 5 mode
Refer to OC1M description.

Bit 3 **OC5PE**: Output compare 5 preload enable
Refer to OC1PE description.

Bit 2 **OC5FE**: Output compare 5 fast enable
Refer to OC1FE description.

Bits 1:0 Reserved, must be kept at reset value.

23.4.25 TIM1 capture/compare register 5 (TIM1_CCR5)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group Channel 5 and Channel 3
Distortion on Channel 3 output:
0: No effect of OC5REF on OC3REFC
1: OC3REFC is the logical AND of OC3REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group Channel 5 and Channel 2
Distortion on Channel 2 output:
0: No effect of OC5REF on OC2REFC
1: OC2REFC is the logical AND of OC2REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group Channel 5 and Channel 1
Distortion on Channel 1 output:
0: No effect of OC5REF on OC1REFC5
1: OC1REFC is the logical AND of OC1REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value
CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC5 output.

23.4.26 TIM1 capture/compare register 6 (TIM1_CCR6)

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value
CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC6 output.

23.4.27 TIM1 alternate function option register 1 (TIM1_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	BK CMP2P	BK CMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	BK CMP2E	BK CMP1E	BKINE
rw	rw			rw	rw	rw							rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.
0000: ETR legacy mode
0001: COMP1 output
0010: COMP2 output
Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input polarity is not inverted (active low if BKP=0, active high if BKP=1)
1: COMP2 input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input polarity is not inverted (active low if BKP=0, active high if BKP=1)
1: COMP1 input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input polarity is not inverted (active low if BKP=0, active high if BKP=1)
1: BKIN input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 136: TIM1 ETR input circuitry and to Figure 157: Break and Break2 circuitry overview.

23.4.28 TIM1 Alternate function register 2 (TIM1_AF2)

Address offset: 0x64

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BK2 CMP2 P	BK2 CMP1 P	BK2 INP	Res.	Res.	Res.	Res.	Res.	Res.	BK2 CMP2E	BK2 CMP1E	BK2INE
				rw	rw	rw							rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: COMP2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP1 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: COMP1 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: BKIN2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BK2CMP2E**: BRK2 COMP2 enable

This bit enables the COMP2 for the timer's BRK2 input. COMP2 output is 'ORed' with the other BRK2 sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: BRK2 COMP1 enable

This bit enables the COMP1 for the timer's BRK2 input. COMP1 output is 'ORed' with the other BRK2 sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: BRK2 BKIN input enable

This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.

0: BKIN2 input disabled

1: BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 157: Break and Break2 circuitry overview.

23.4.29 TIM1 timer input selection register (TIM1_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: selects TI4[0] to TI4[15] input
0000: TIM1_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: selects TI3[0] to TI3[15] input
0000: TIM1_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input
0000: TIM1_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM1_CH1 input
Others: Reserved

23.4.30 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 169. TIM1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM1_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM1_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	**TIM1_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value											0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**TIM1_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**TIM1_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value															0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**TIM1_EGR**	Res.	Res.	Res	Res	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	**TIM1_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR1 Input Capture mode	Res.	Res.	Res	Res	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIM1_CCMR2** Output Compare mode	Res.	Res.	Res	Res	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM1_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 169. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**TIM1_CNT**	UIFCP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM1_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM1_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM1_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**TIM1_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM1_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIM1_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIM1_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIM1_BDTR**	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIM1_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIM1_DMAR**	DMAB[31:0]																															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM1_OR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP	Res.	Res.	TIM1_ETR_ADC_RMP	
	Reset value																												0			0	0

Table 169. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54	**TIM1_CCMR3** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x58	**TIM1_CCR5**	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]															
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	**TIM1_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**TIM1_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]						BKCMP2P	BKCMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value															0	0	0	0			0	0	0							0	0	1
0x64	**TIM1_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP2P	BK2CMP1P	BK2INP	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value																					0	0	0							0	0	1
0x68	**TIM1_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0

Refer to *Section 2.4 on page 62* for the register boundary addresses.

24 General-purpose timer (TIM2)

24.1 TIM2 introduction

The general-purpose timer TIM2 consists of a 32-bit auto-reload counter driven by a programmable prescaler.

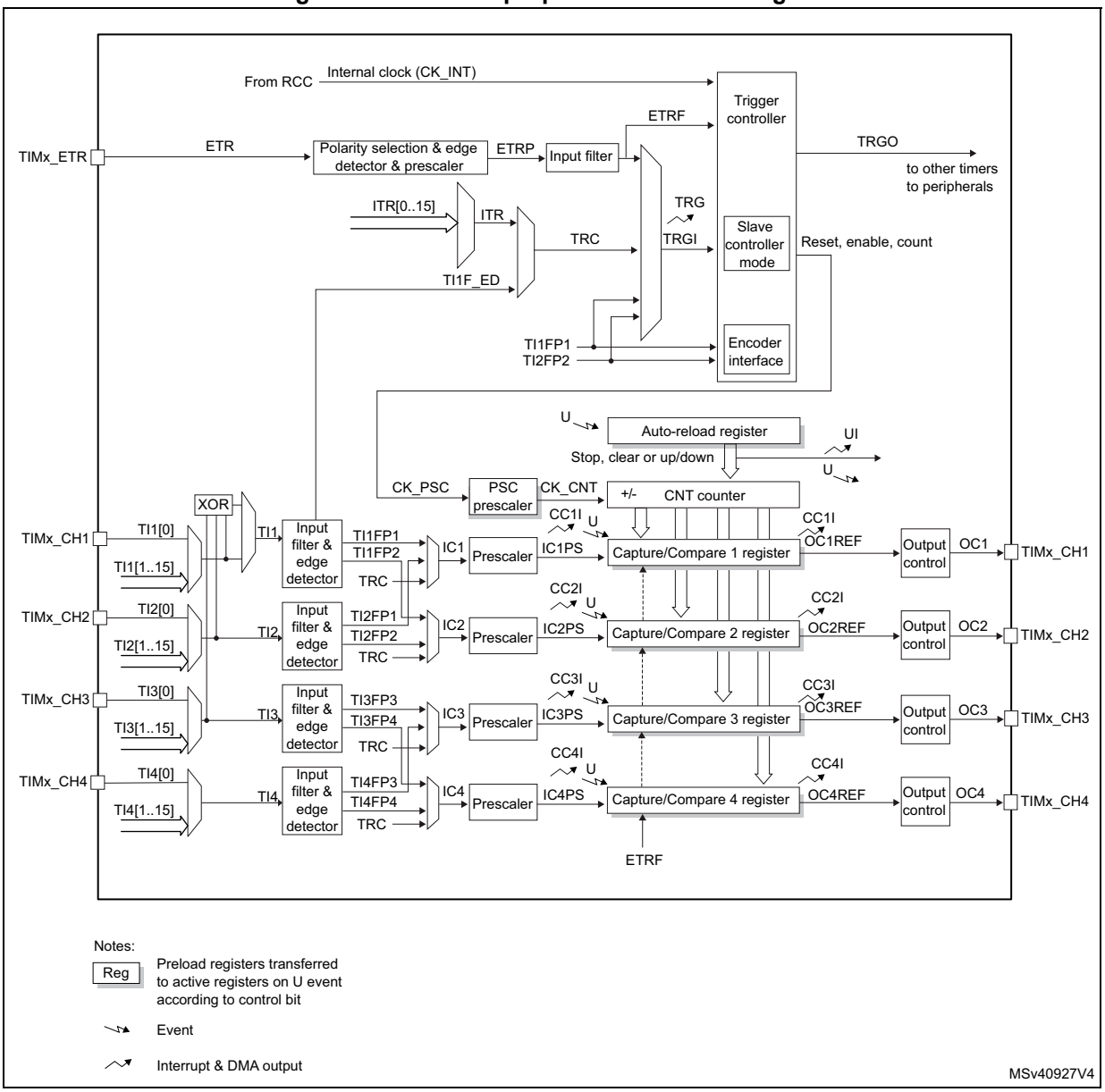
The timer may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

24.2 TIM2 main features

- 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 174. General-purpose timer block diagram



24.3 TIM2 functional description

24.3.1 Time-base unit

The main block of the programmable timer is a 32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 175 and *Figure 176* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 175. Counter timing diagram with prescaler division change from 1 to 2

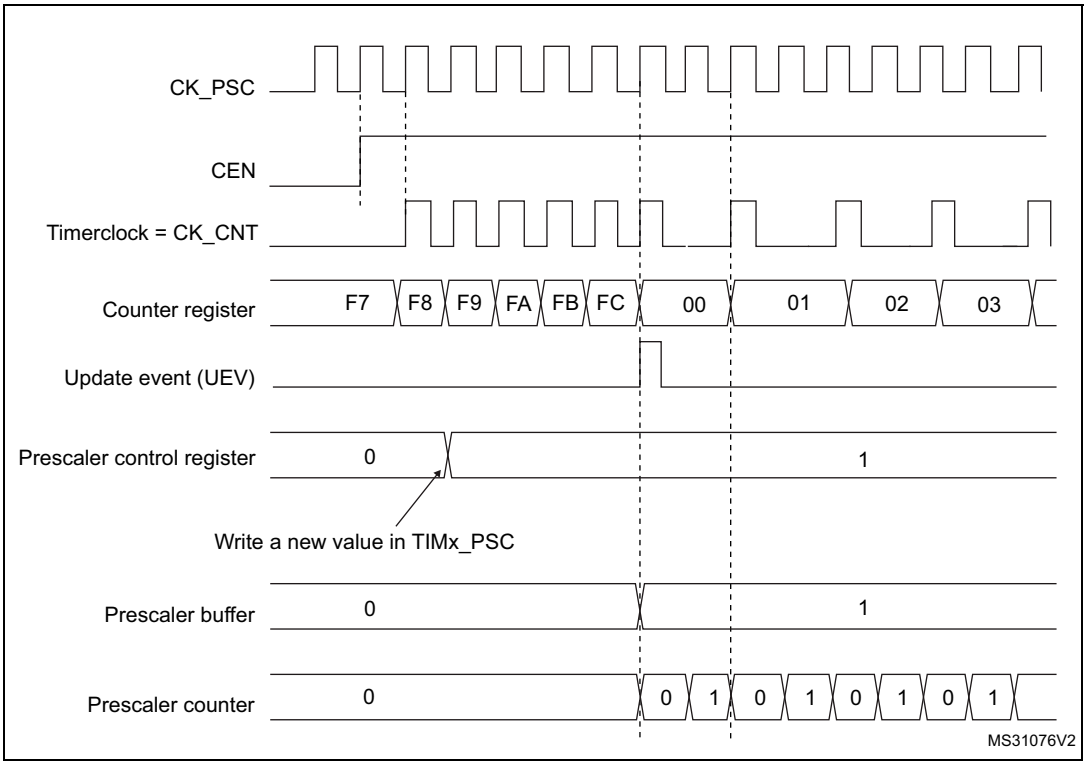
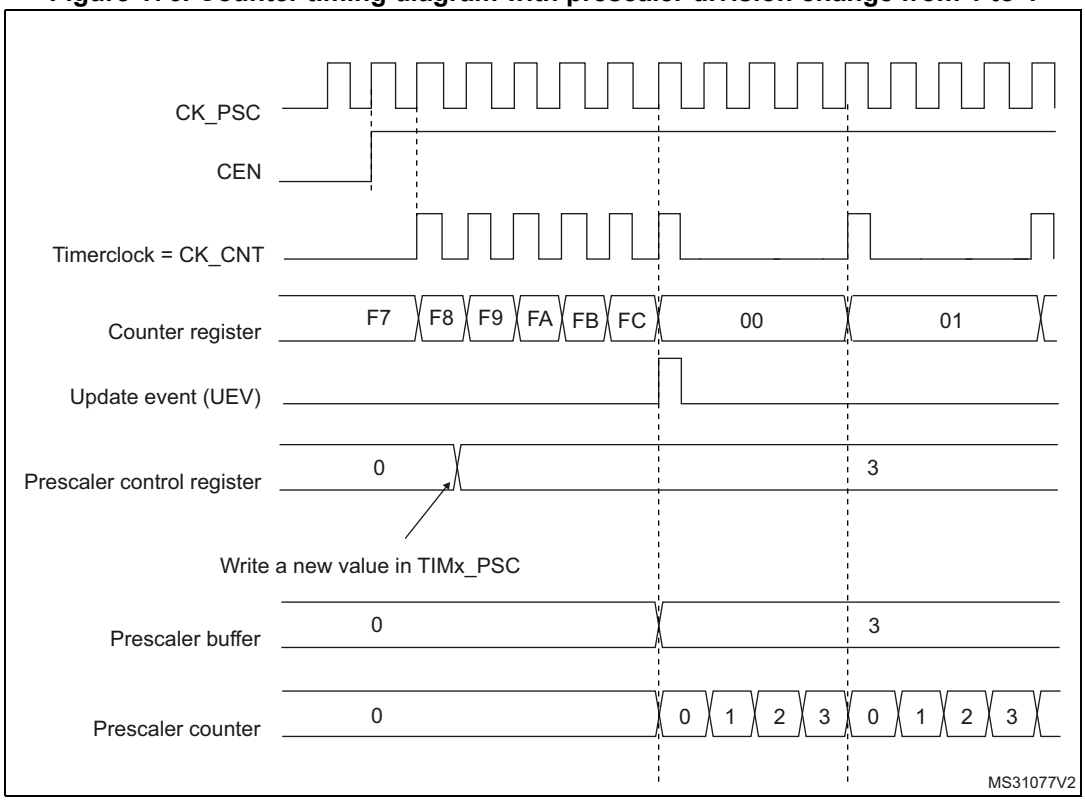


Figure 176. Counter timing diagram with prescaler division change from 1 to 4



24.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 177. Counter timing diagram, internal clock divided by 1

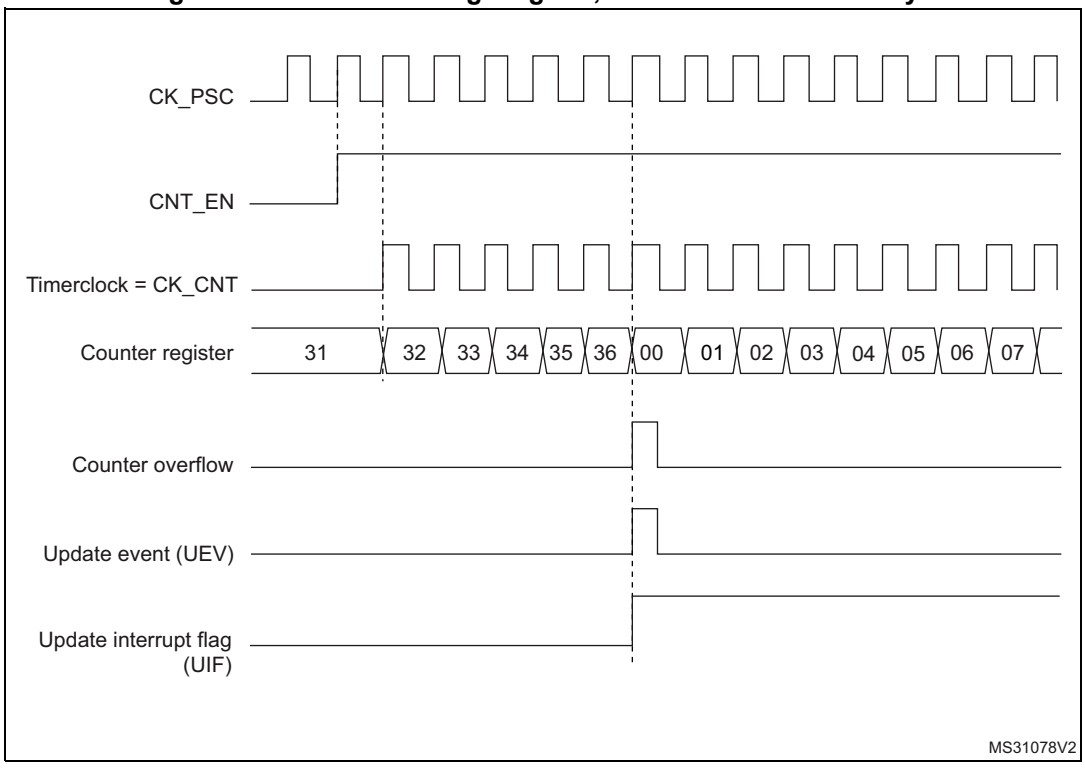


Figure 178. Counter timing diagram, internal clock divided by 2

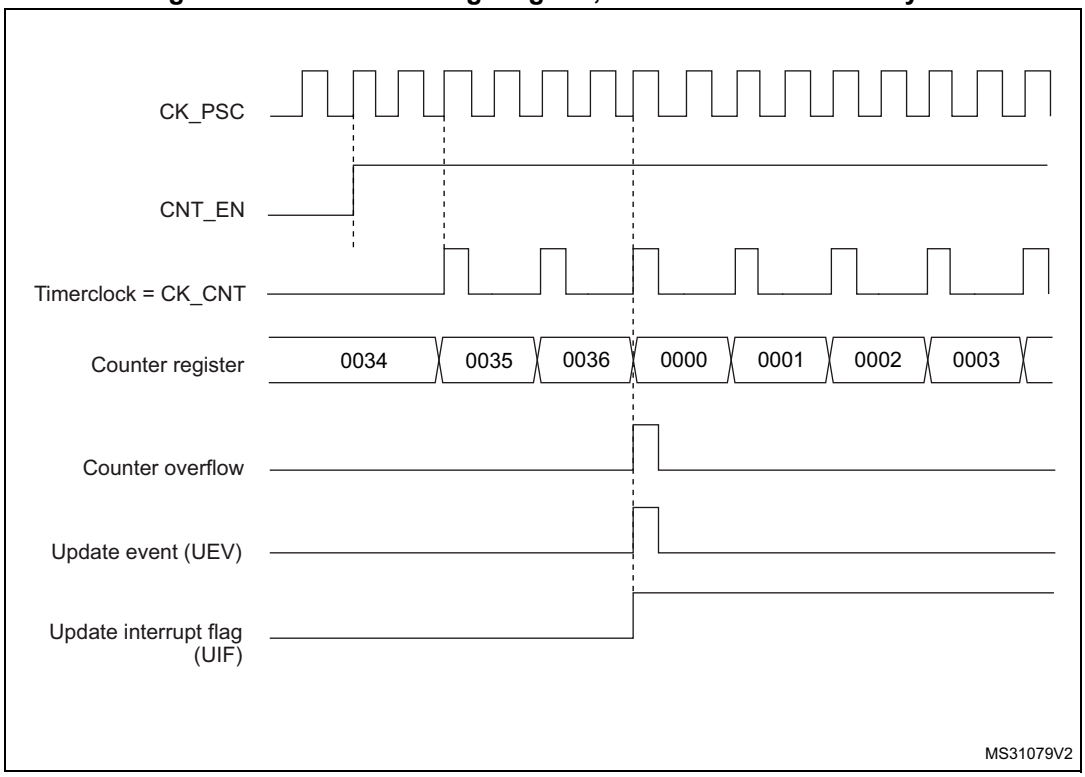


Figure 179. Counter timing diagram, internal clock divided by 4

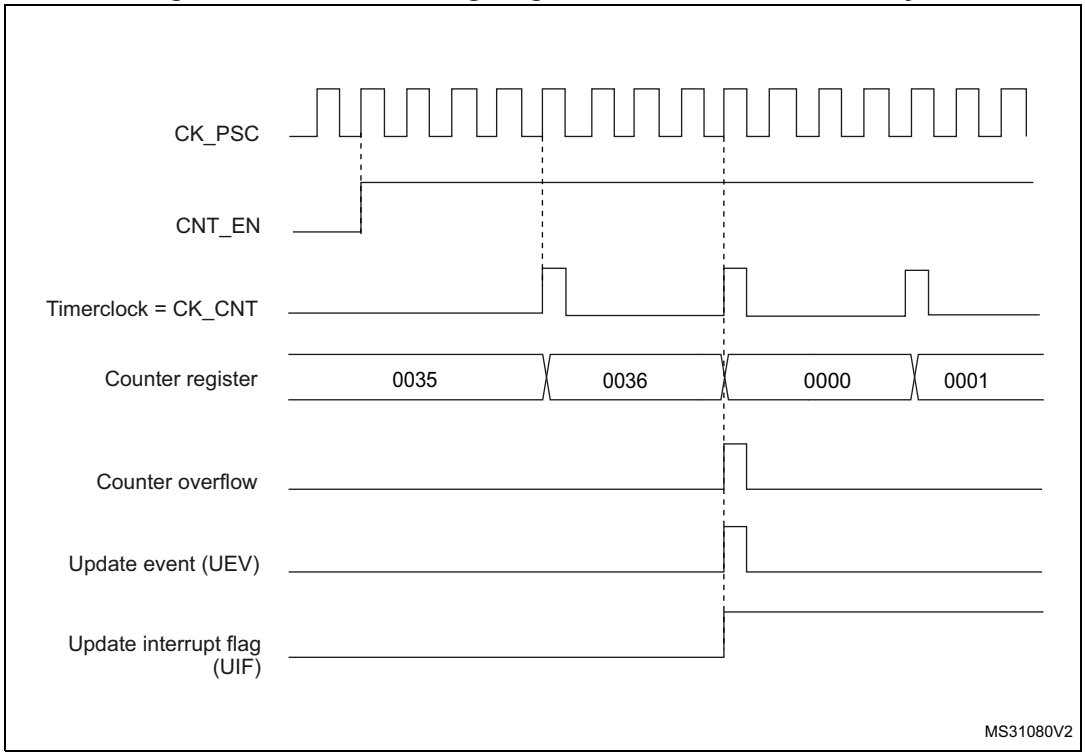


Figure 180. Counter timing diagram, internal clock divided by N

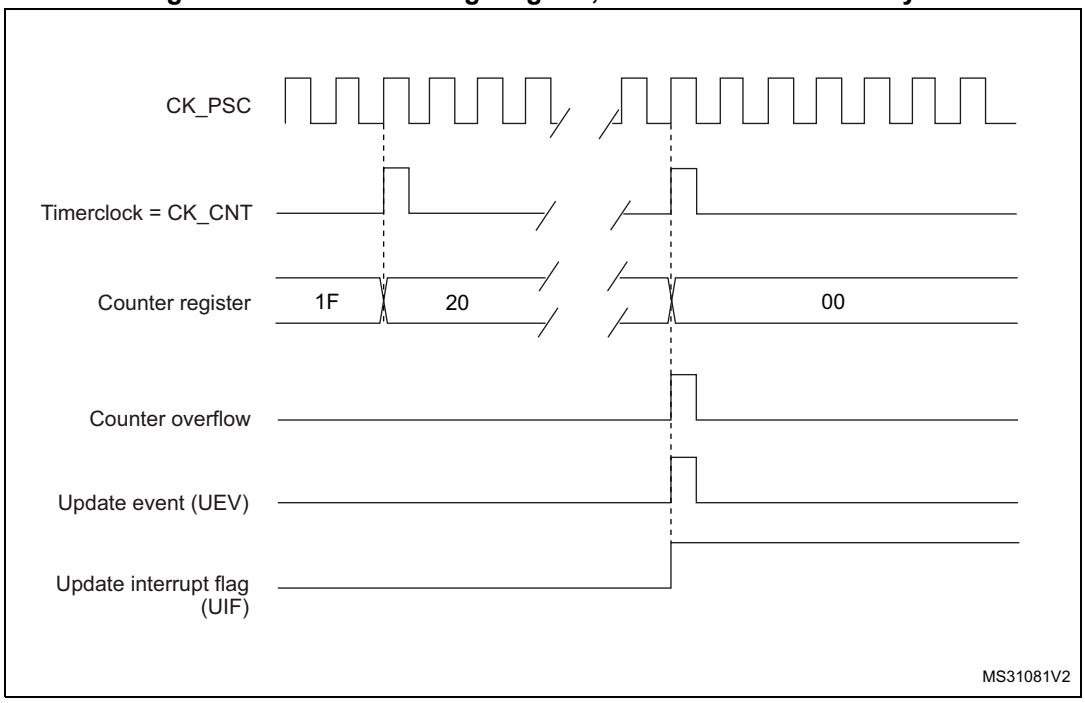


Figure 181. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

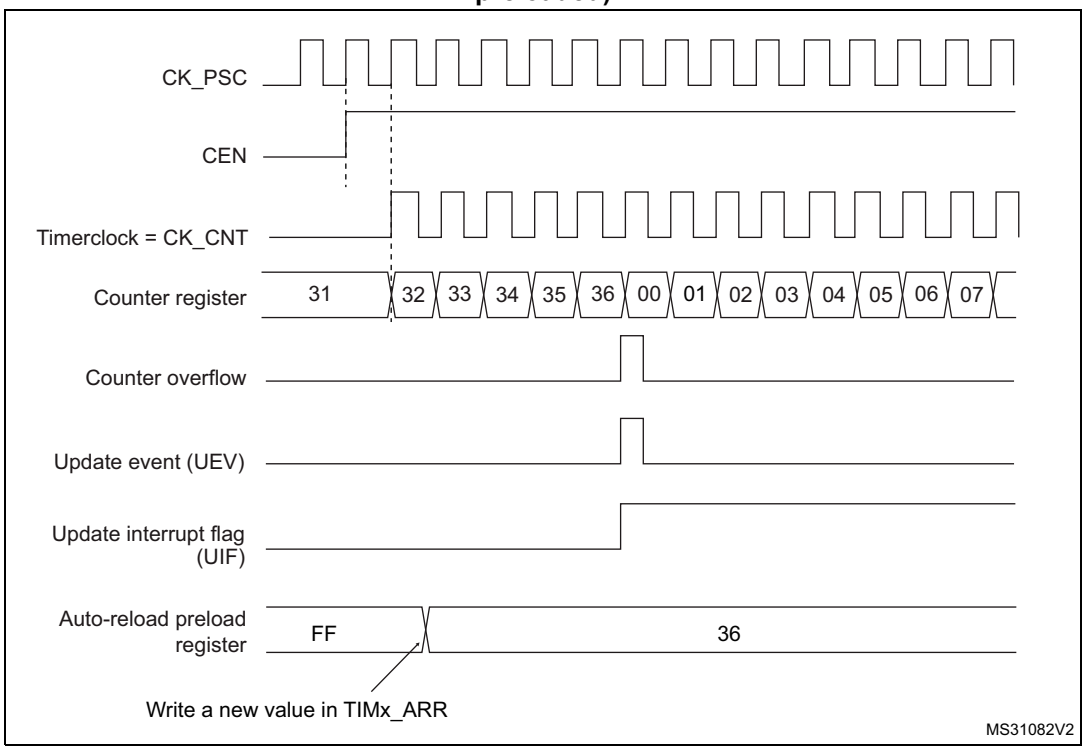
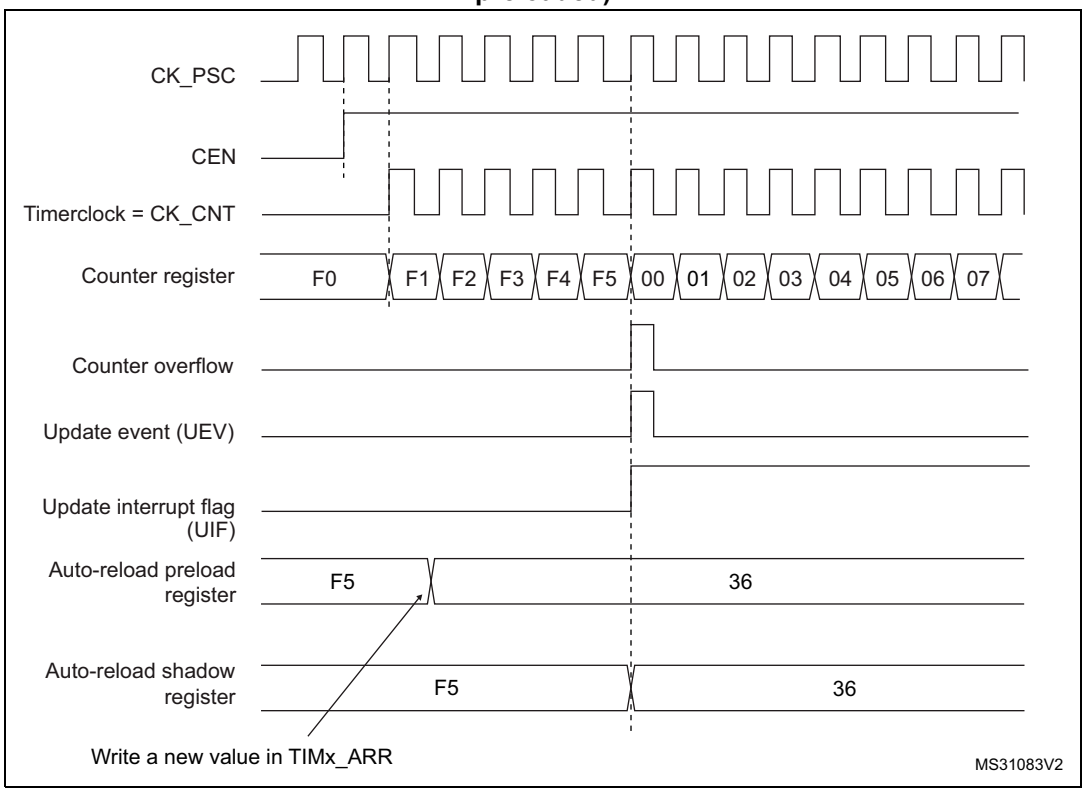


Figure 182. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generate at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 183. Counter timing diagram, internal clock divided by 1

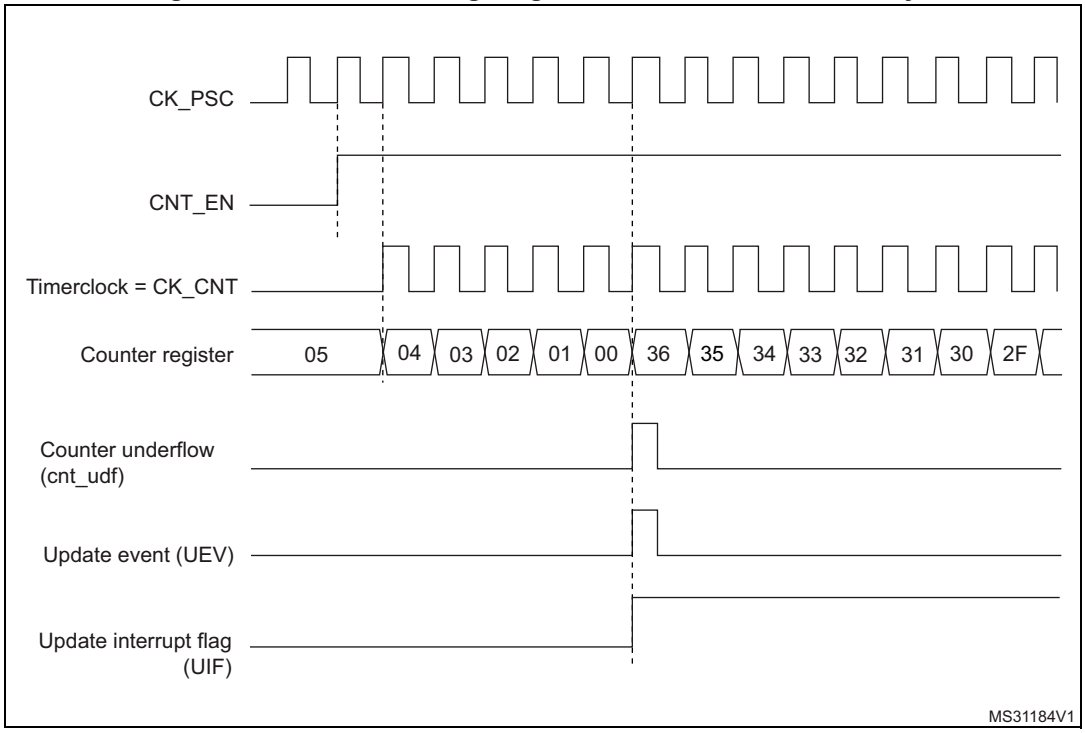


Figure 184. Counter timing diagram, internal clock divided by 2

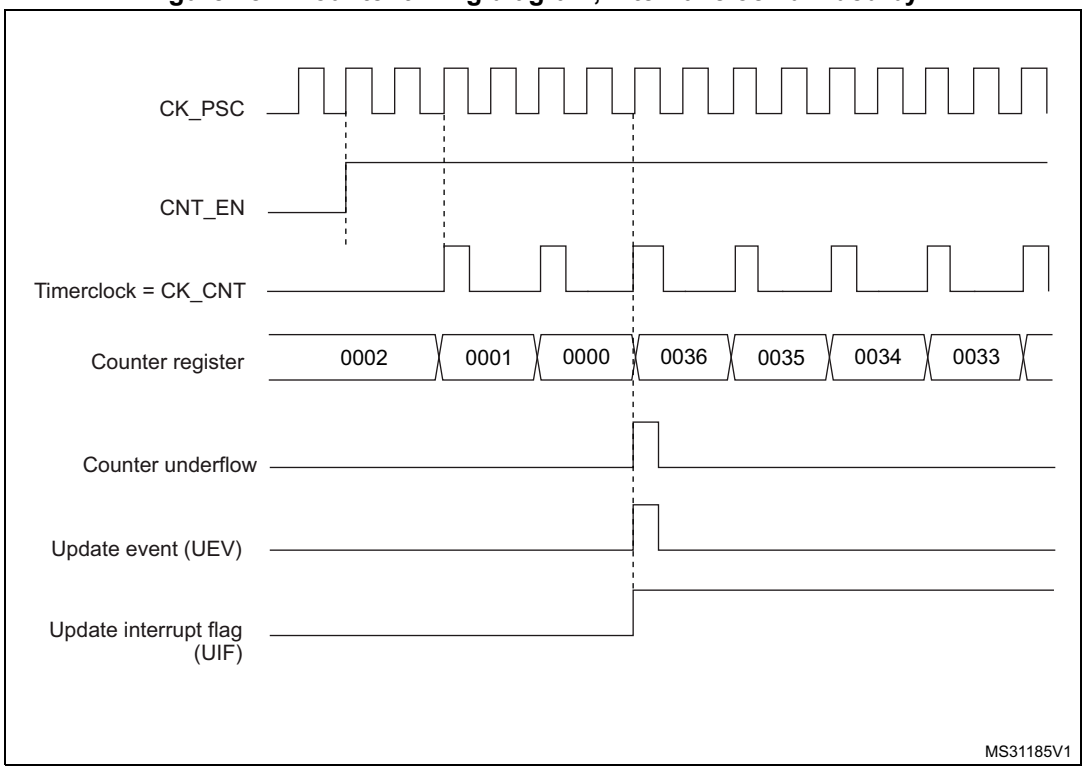


Figure 185. Counter timing diagram, internal clock divided by 4

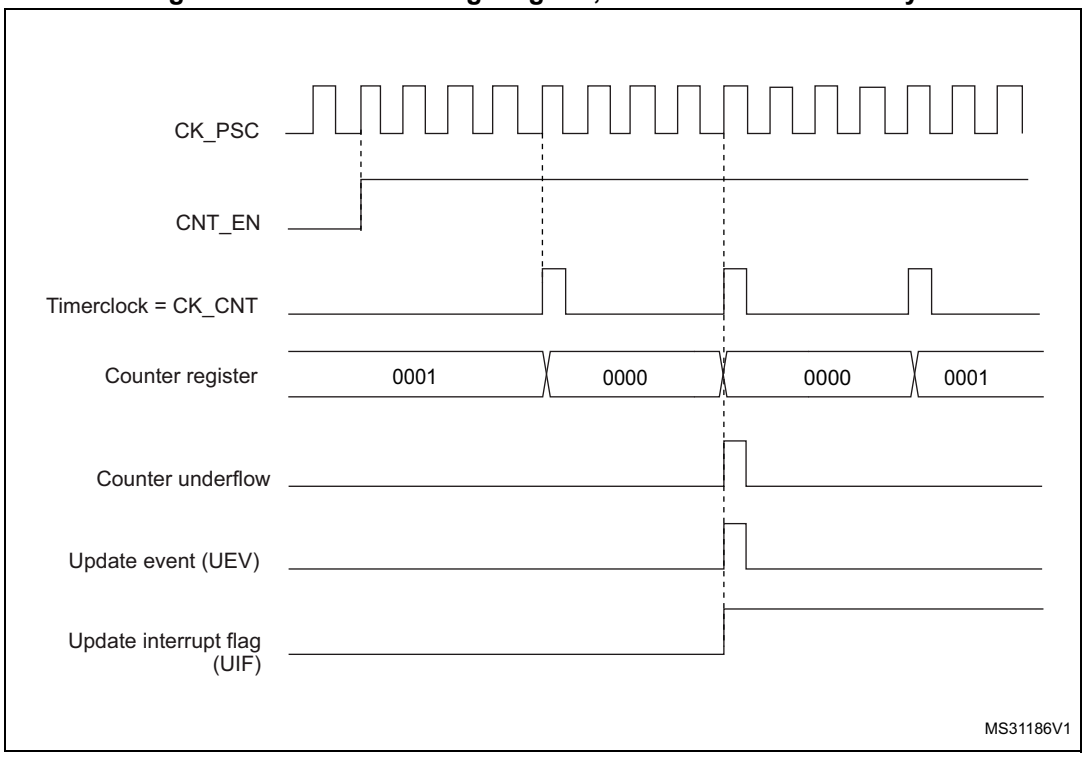


Figure 186. Counter timing diagram, internal clock divided by N

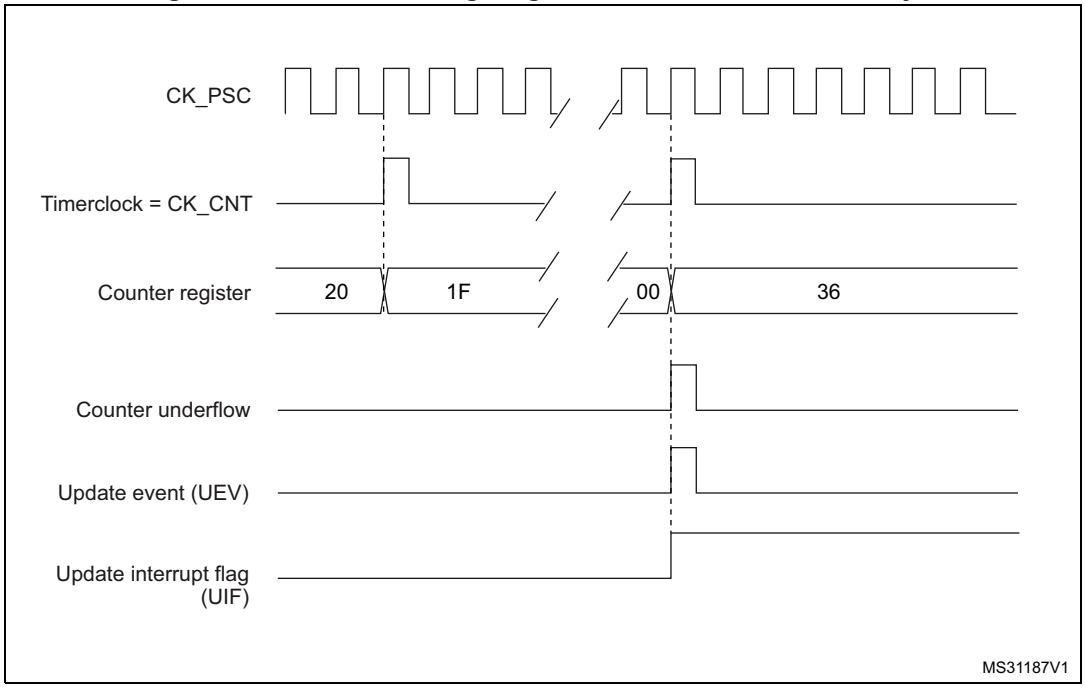
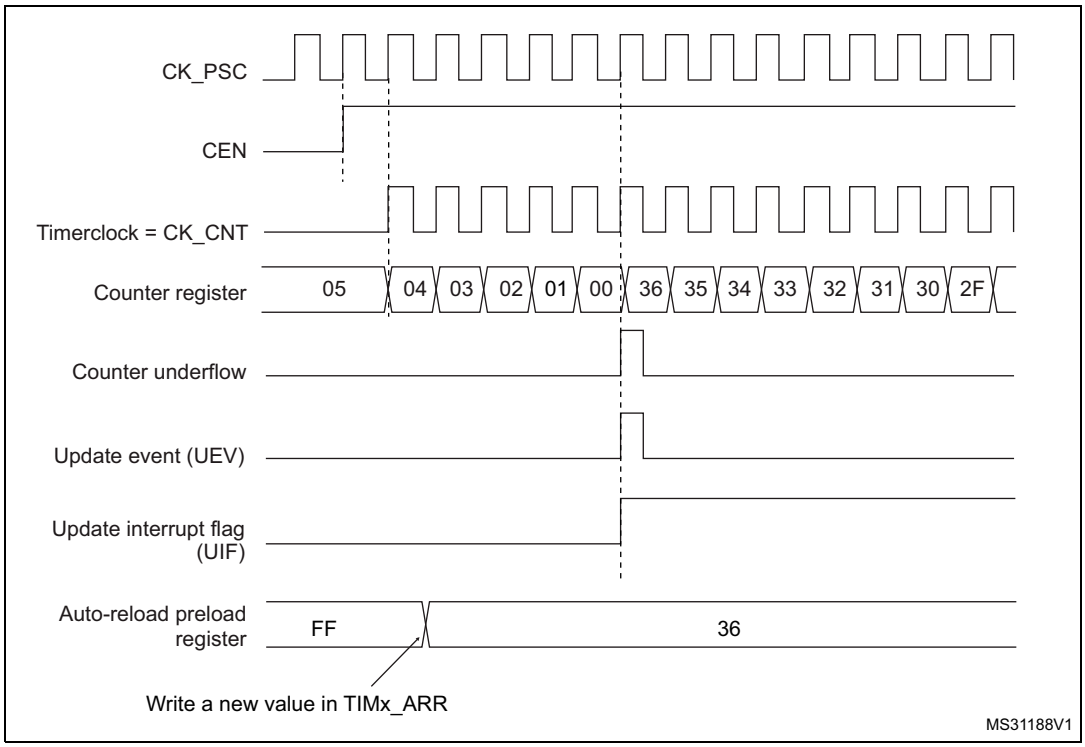


Figure 187. Counter timing diagram, Update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or

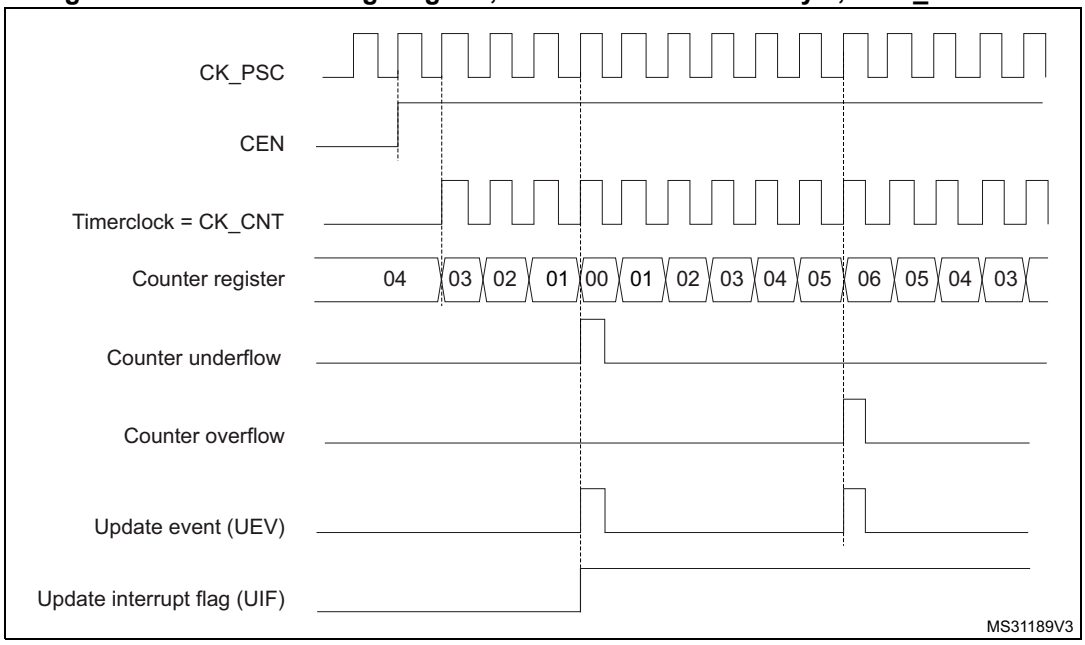
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 188. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 24.4.1: TIM2 control register 1 (TIM2_CR1) on page 763*).

Figure 189. Counter timing diagram, internal clock divided by 2

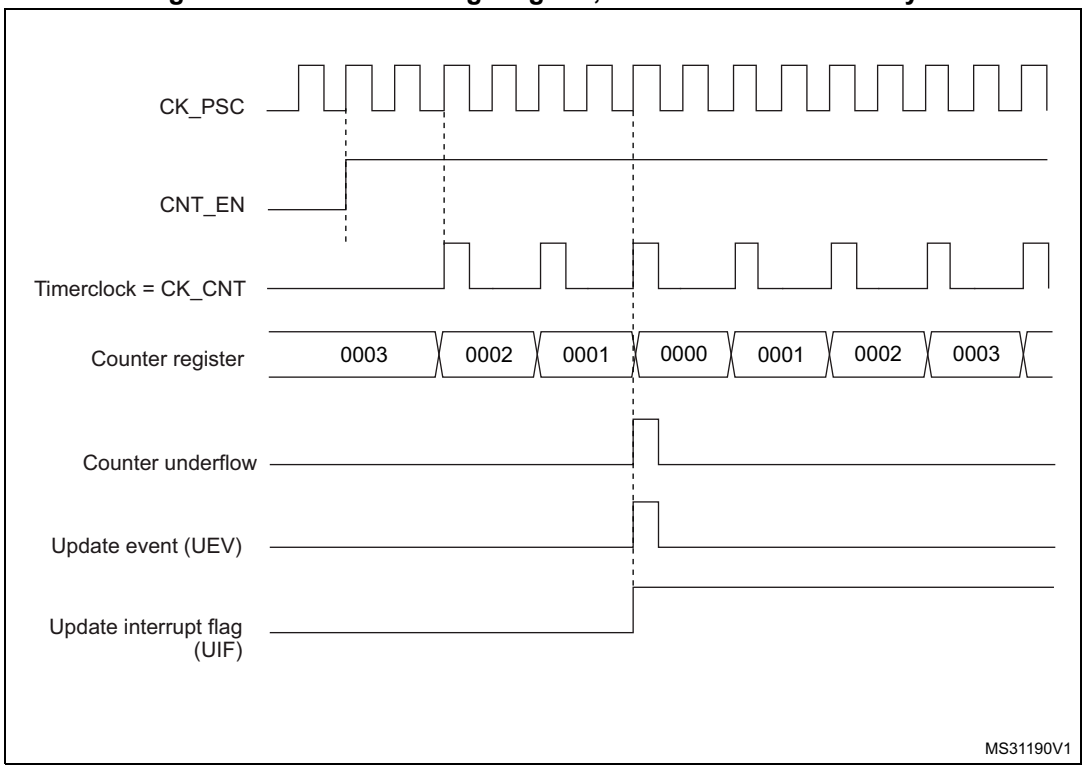
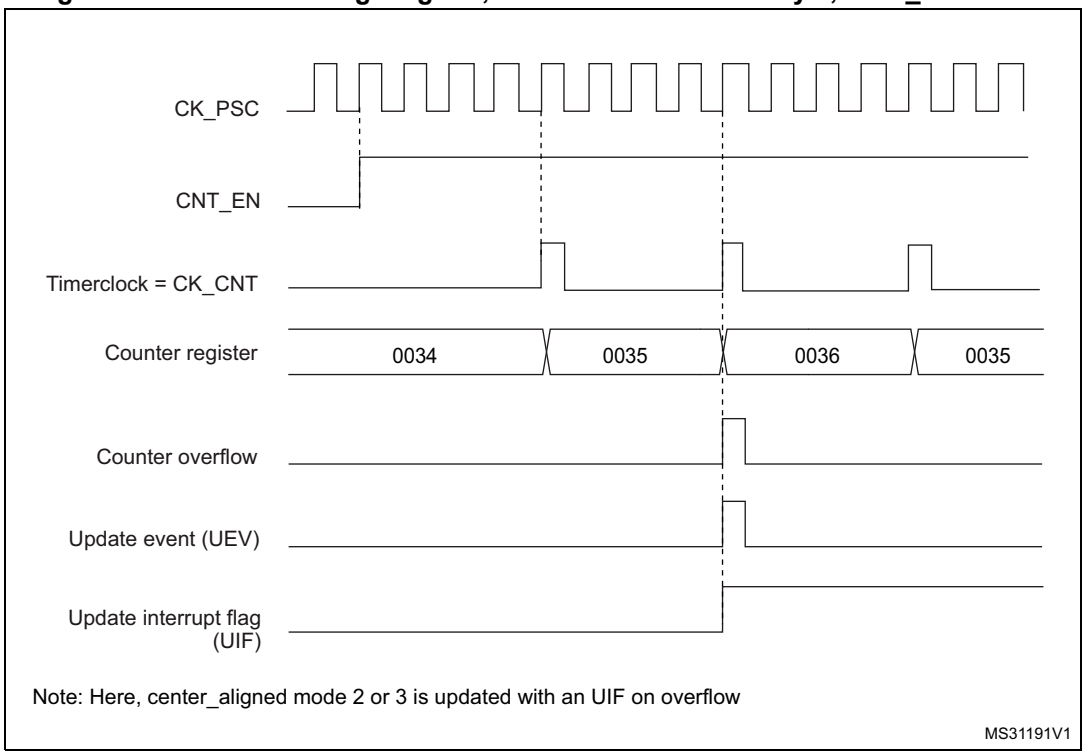


Figure 190. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 191. Counter timing diagram, internal clock divided by N

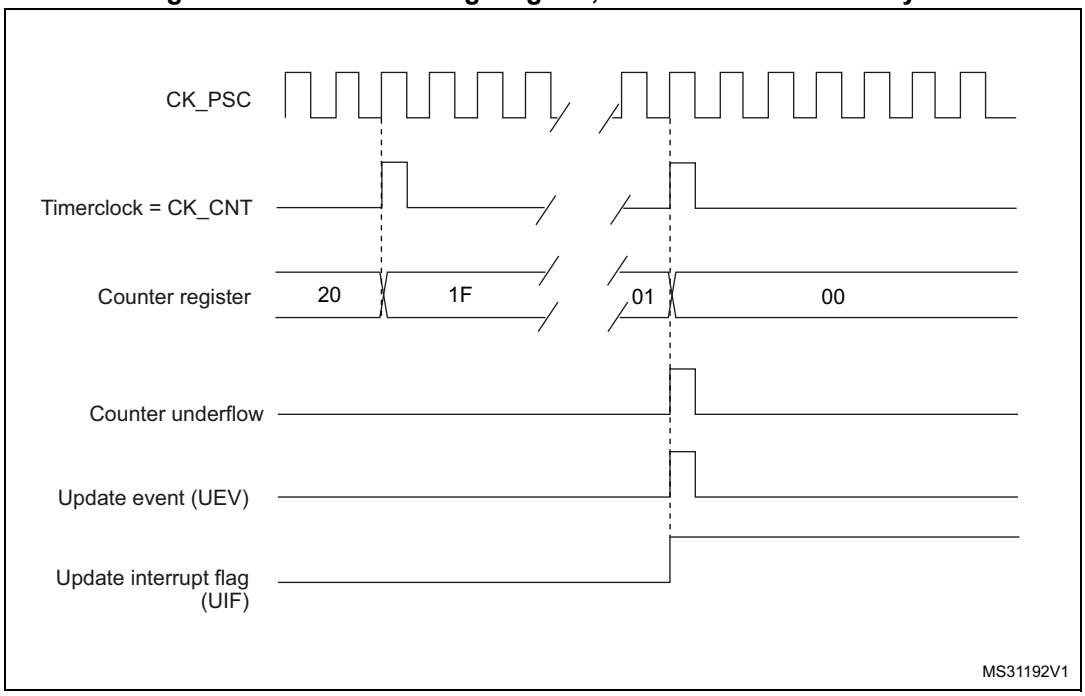


Figure 192. Counter timing diagram, Update event with ARPE=1 (counter underflow)

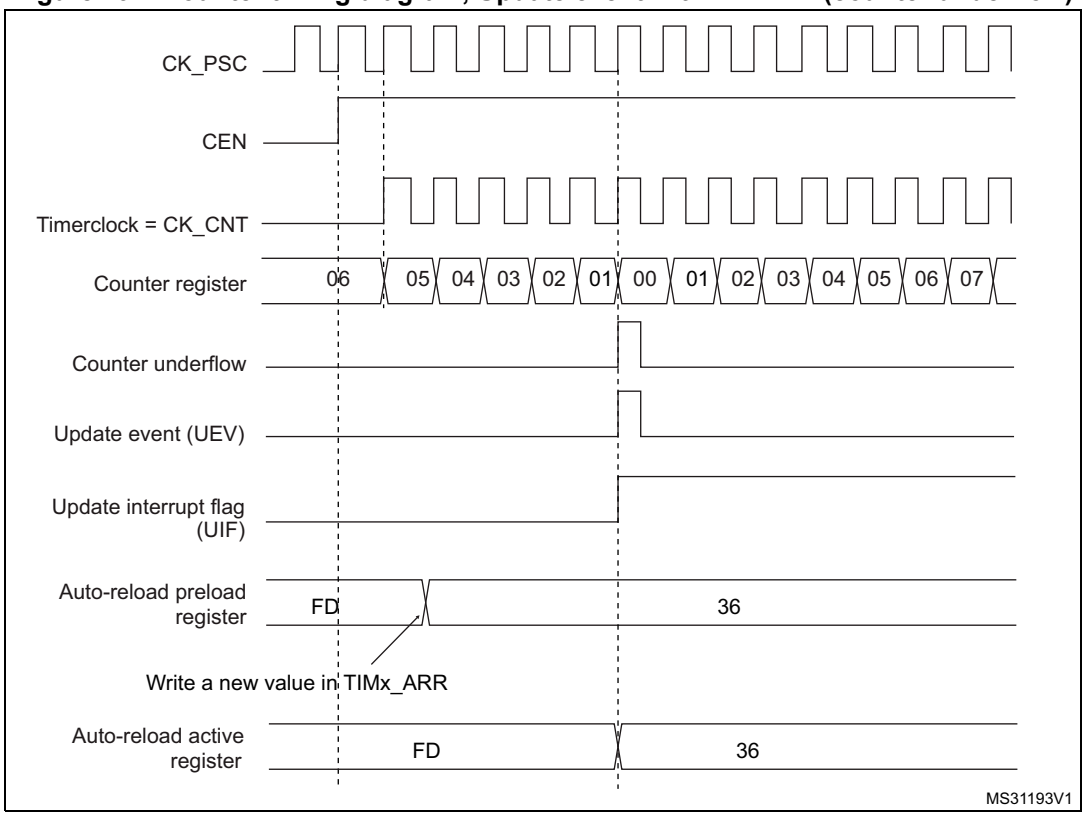
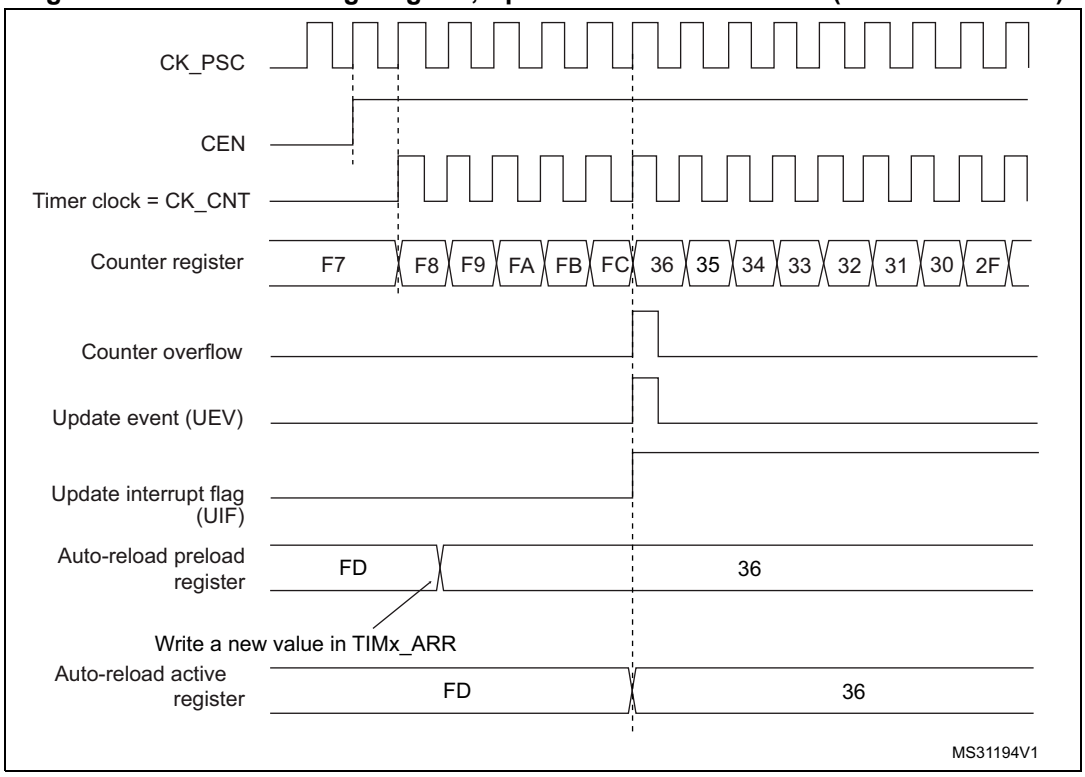


Figure 193. Counter timing diagram, Update event with ARPE=1 (counter overflow)



24.3.3 Clock selection

The counter clock can be provided by the following clock sources:

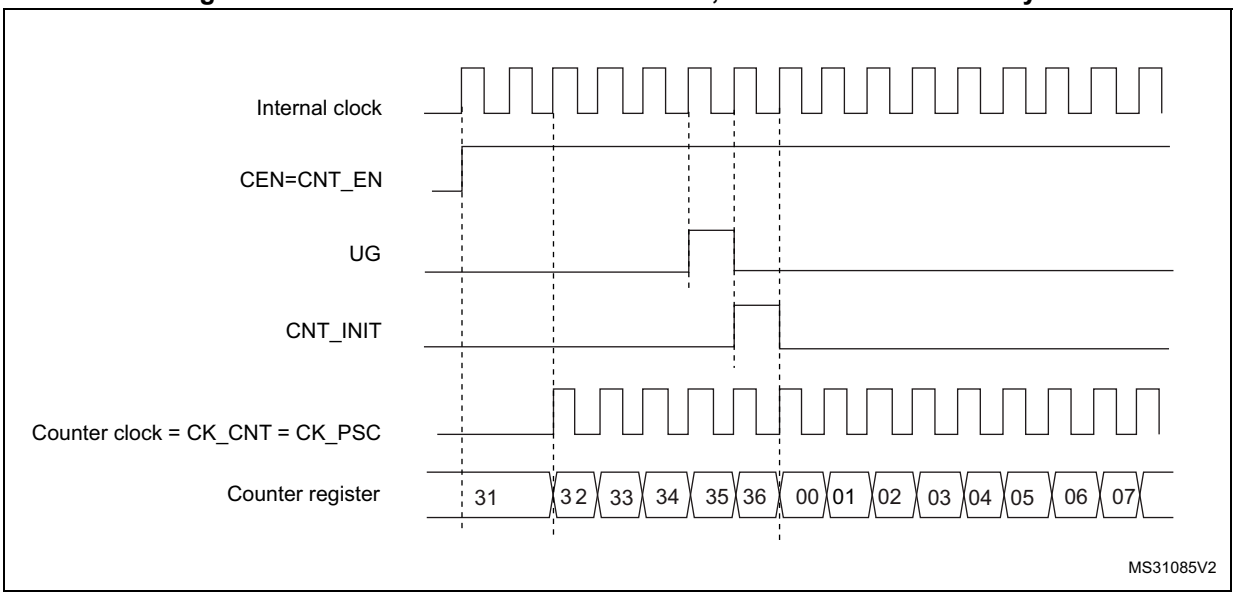
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to *: Using one timer as prescaler for another timer on page 758* for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 194 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

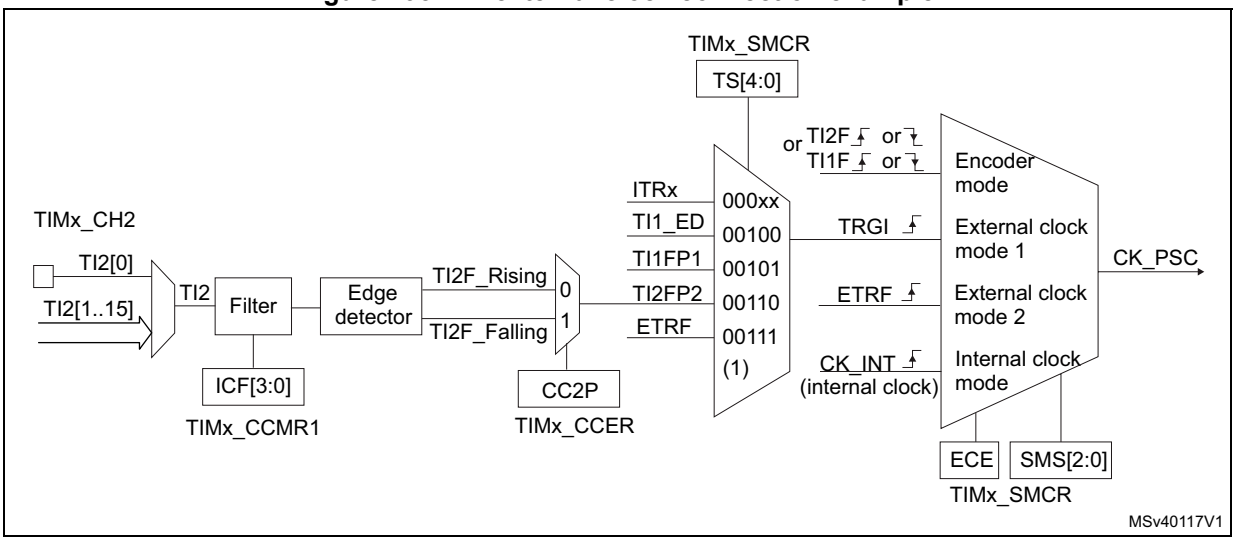
Figure 194. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 195. TI2 external clock connection example



1. Codes ranging from 01000 to 11111: ITRy.

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= ‘01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

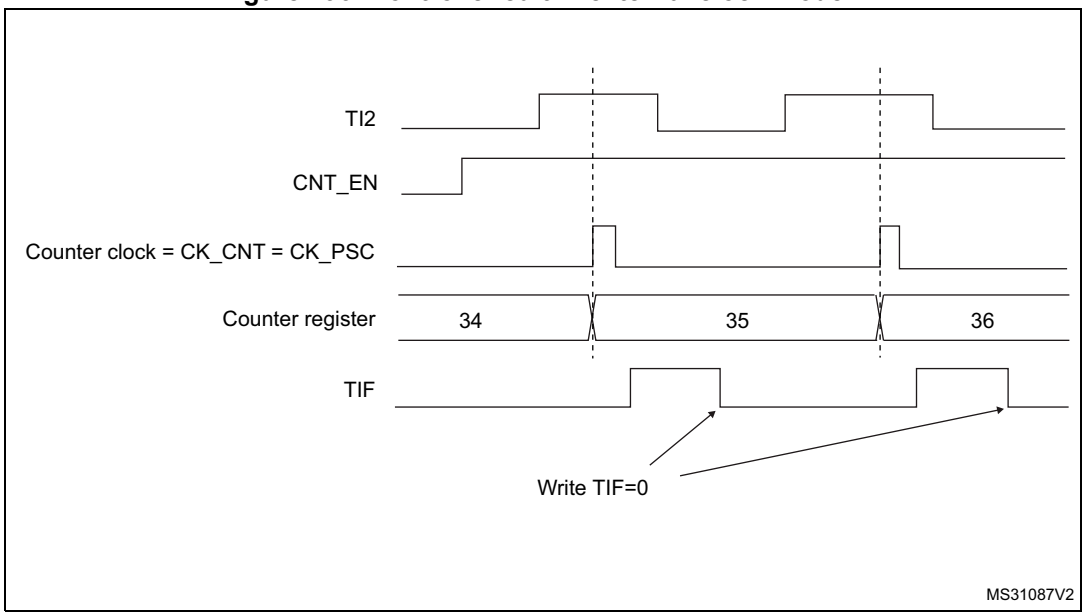
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 196. Control circuit in external clock mode 1



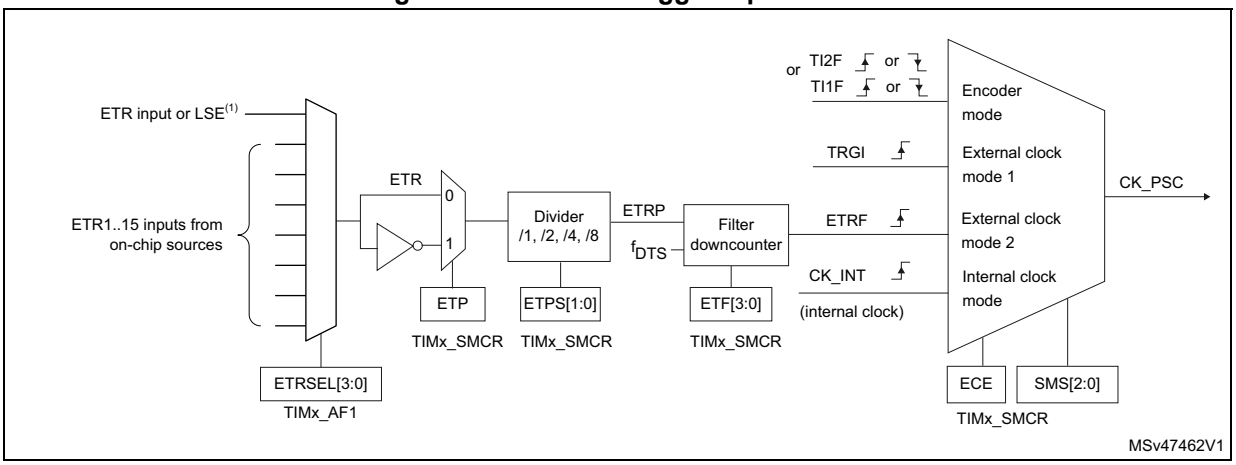
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

Figure 197 gives an overview of the external trigger input block.

Figure 197. External trigger input block



1. As per ETR_RMP bit programming.

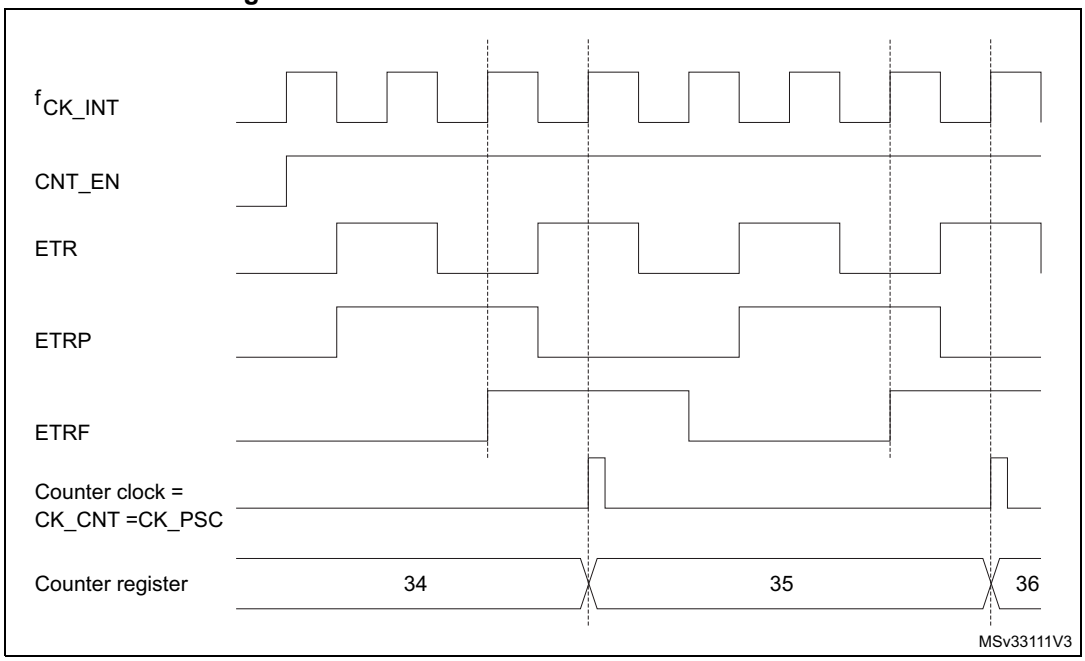
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register and the ETR_RMP bit in the TIM2_OR1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
4. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
5. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.

Figure 198. Control circuit in external clock mode 2



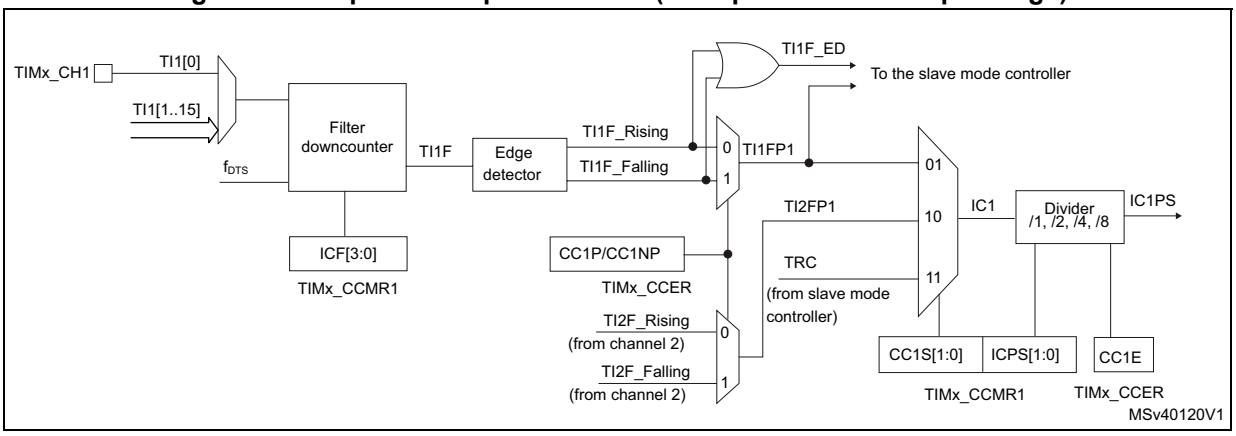
24.3.4 Capture/Compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 199. Capture/Compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 200. Capture/Compare channel 1 main circuit

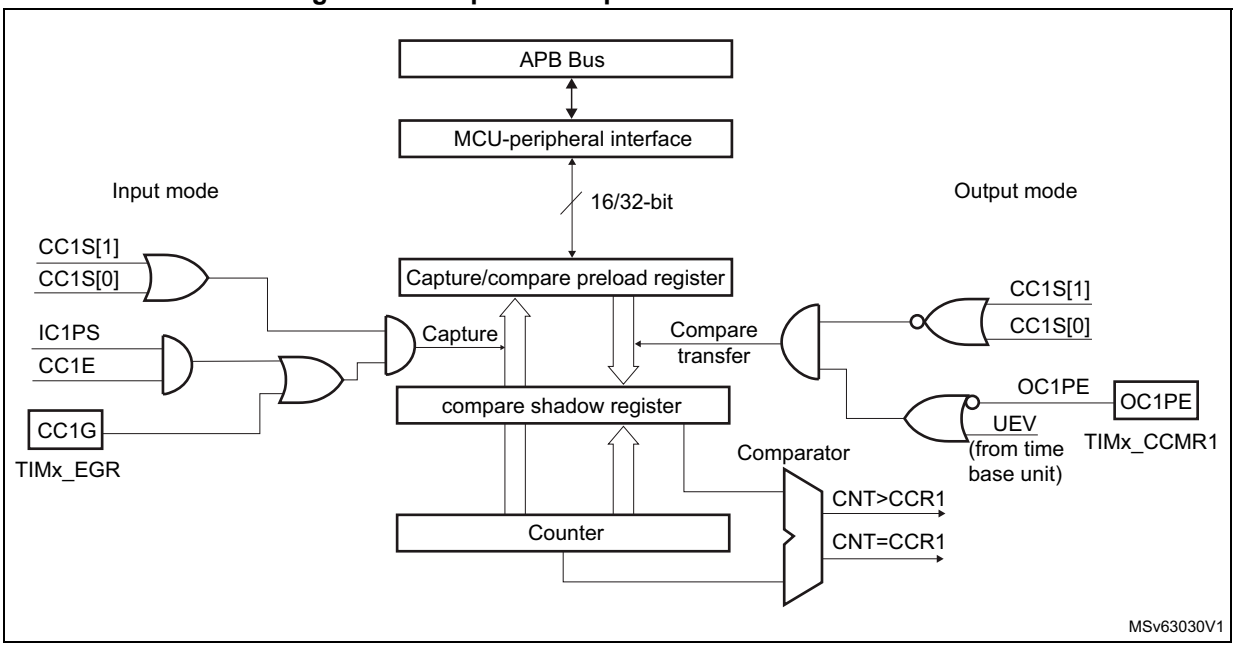
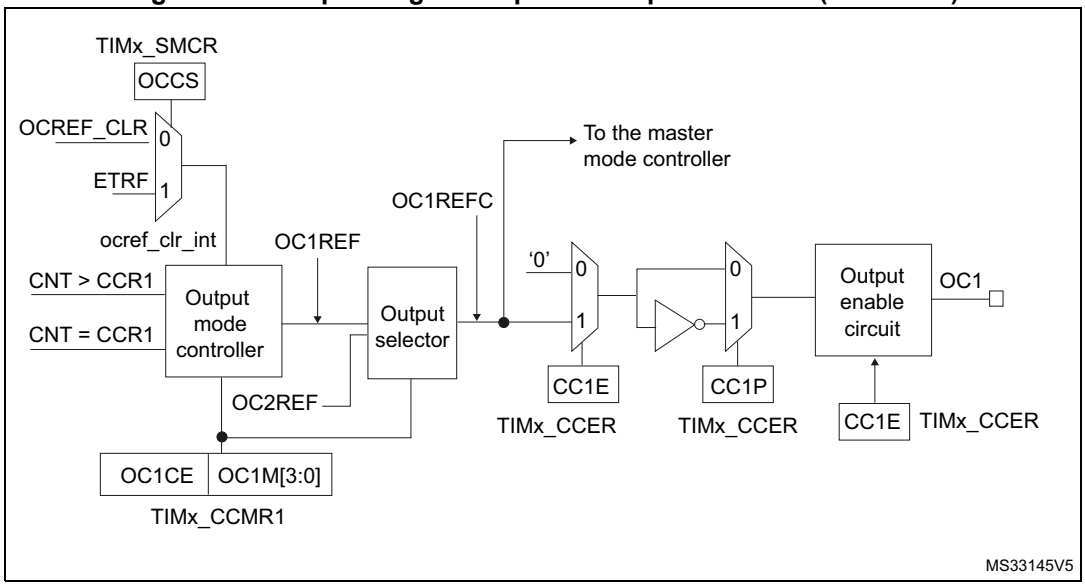


Figure 201. Output stage of Capture/Compare channel (channel 1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

24.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

24.3.6 PWM input mode

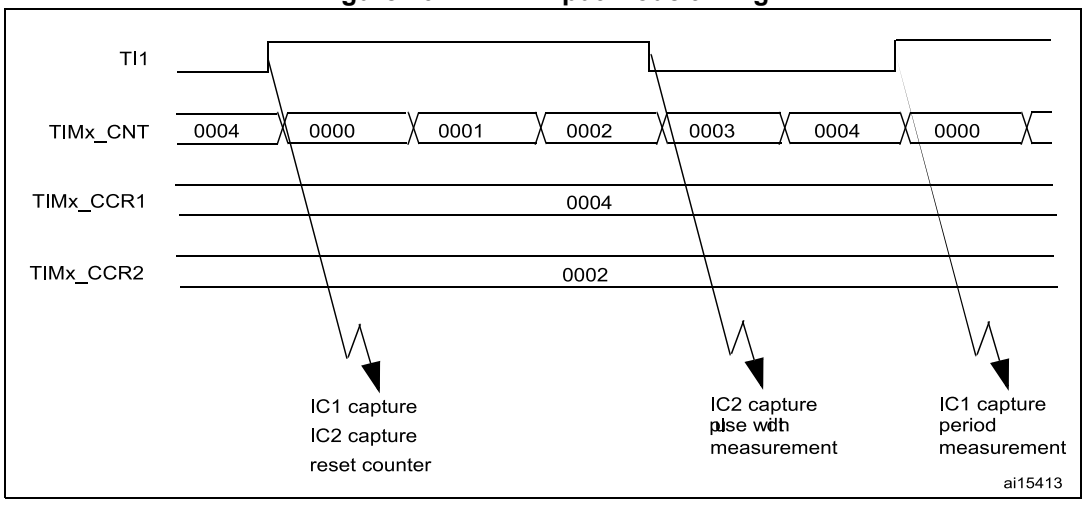
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ‘0’ and the CC1NP bit to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to ‘1’ and the CC2NP bit to ’0’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1 in the TIMx_CCER register.

Figure 202. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

24.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

24.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

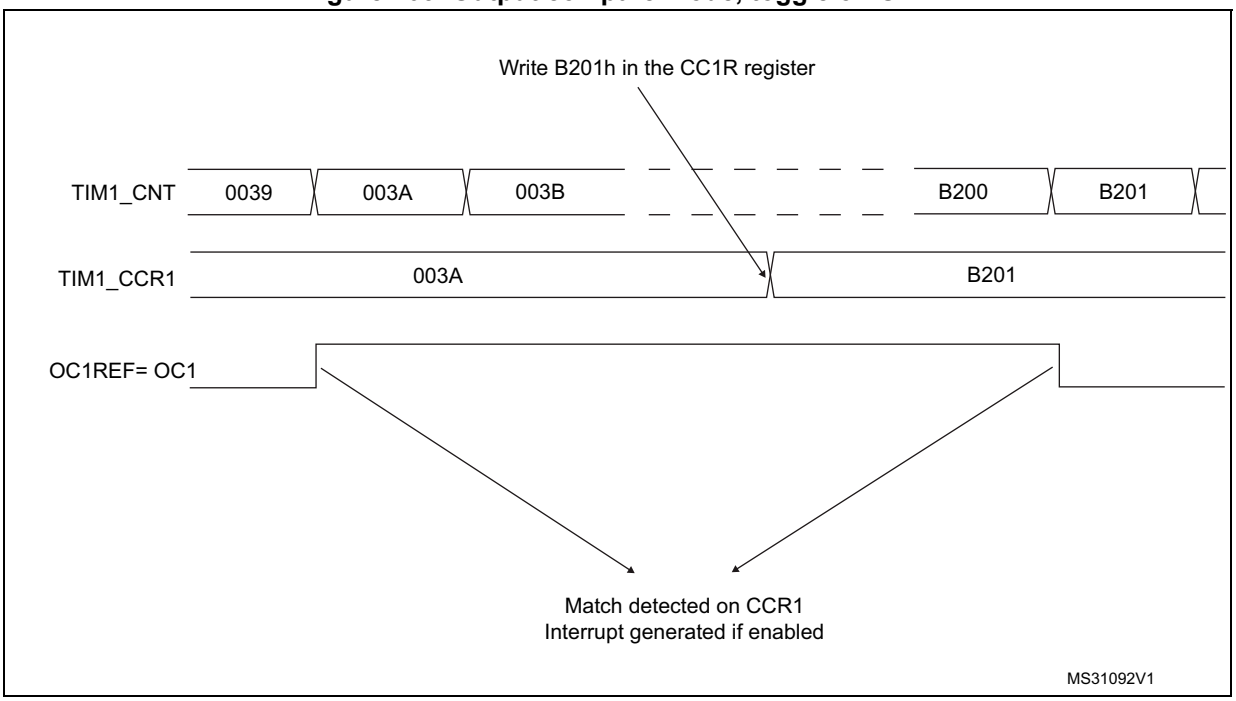
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, one must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 203*.

Figure 203. Output compare mode, toggle on OC1



24.3.9 PWM mode

Pulse width modulation mode permits to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be

cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

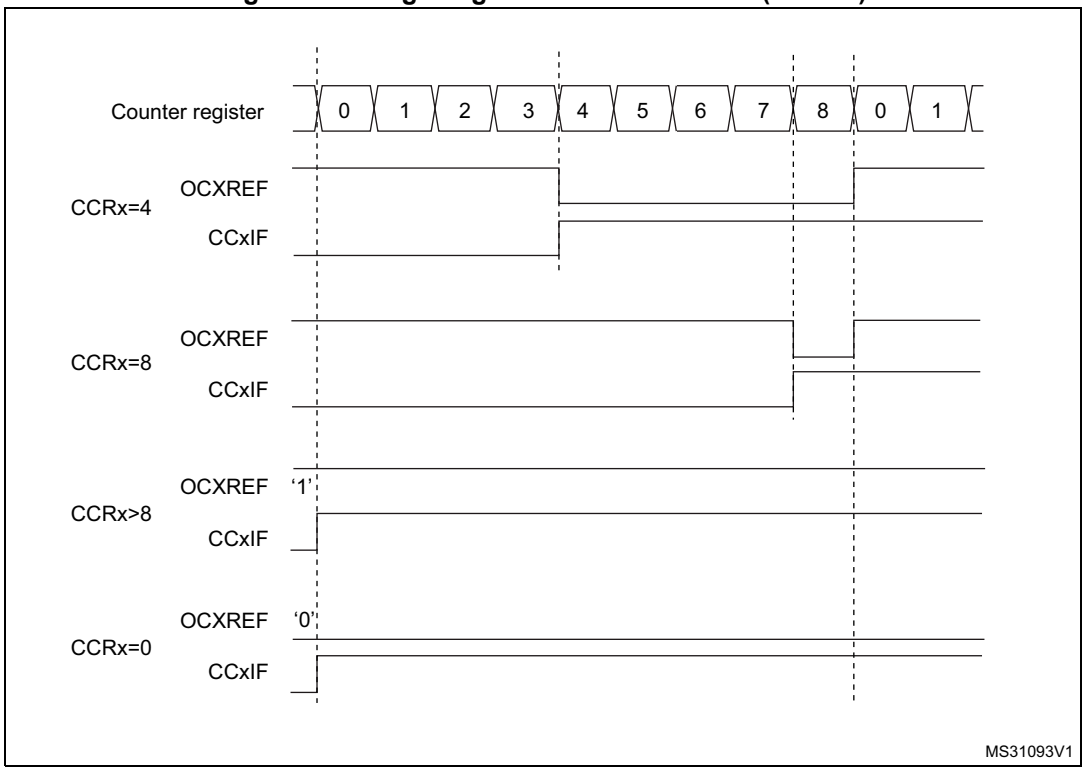
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 723*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1. If the compare value is 0 then OCxREF is held at ‘0. *Figure 204* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 204. Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode on page 726*.

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at 100%. PWM is not possible in this mode.

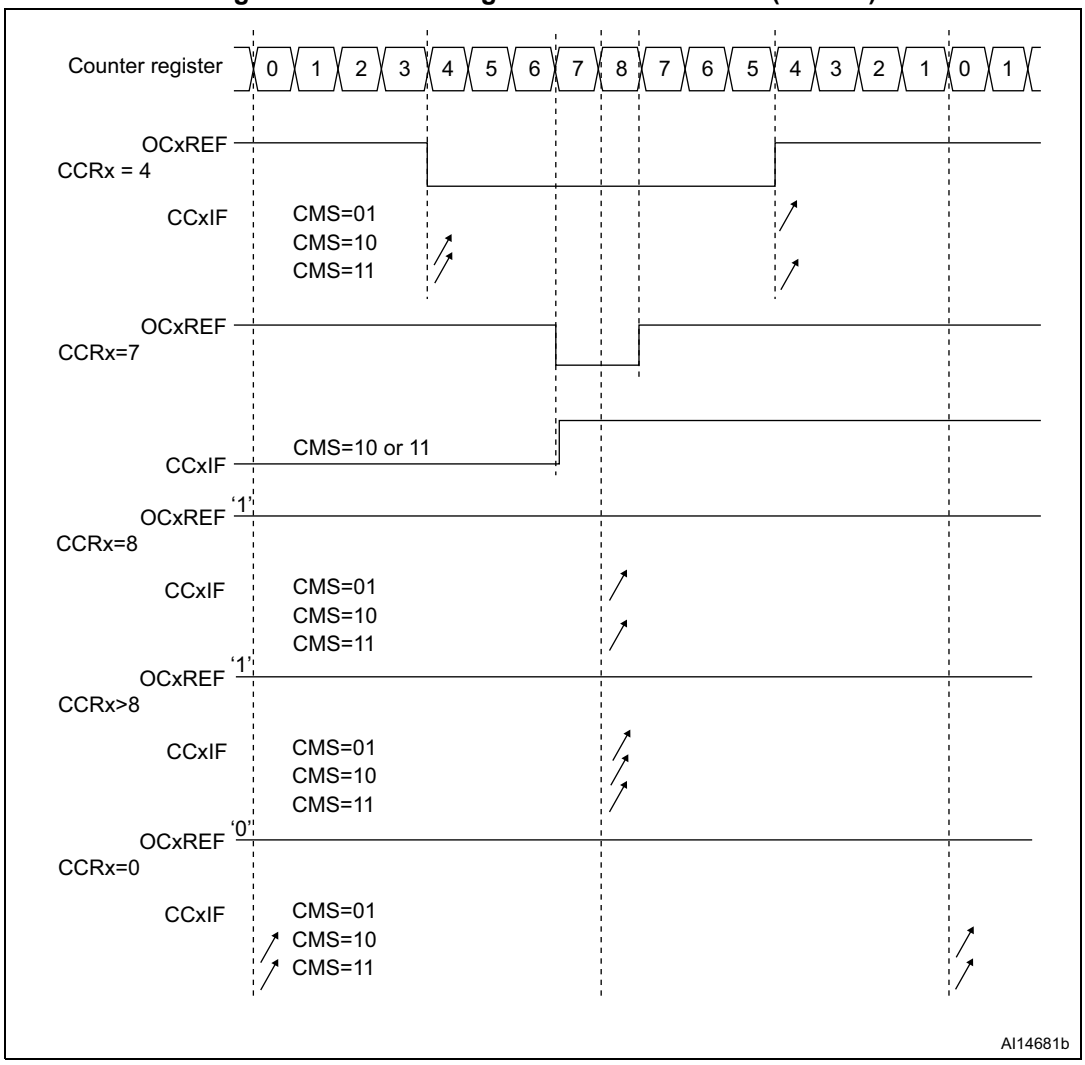
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting) on page 729*.

Figure 205 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 205. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

24.3.10 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

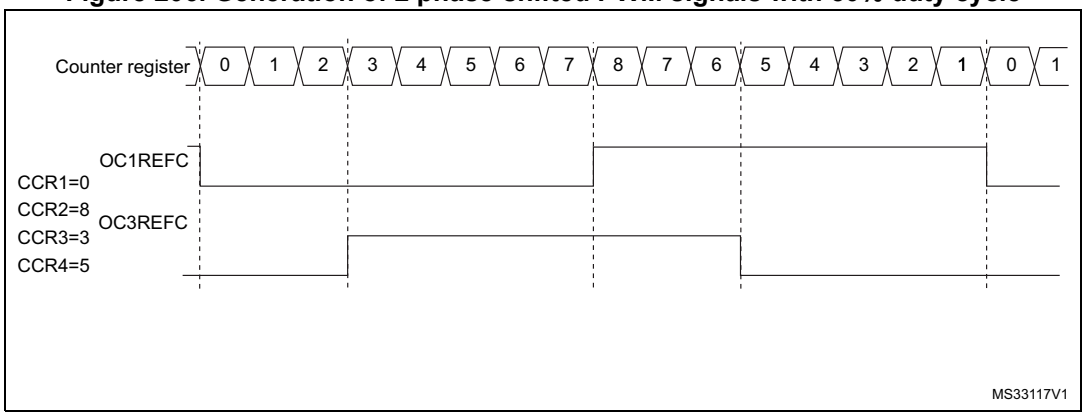
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 2.

Figure 206 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

Figure 206. Generation of 2 phase-shifted PWM signals with 50% duty cycle



24.3.11 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

– OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2

– OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

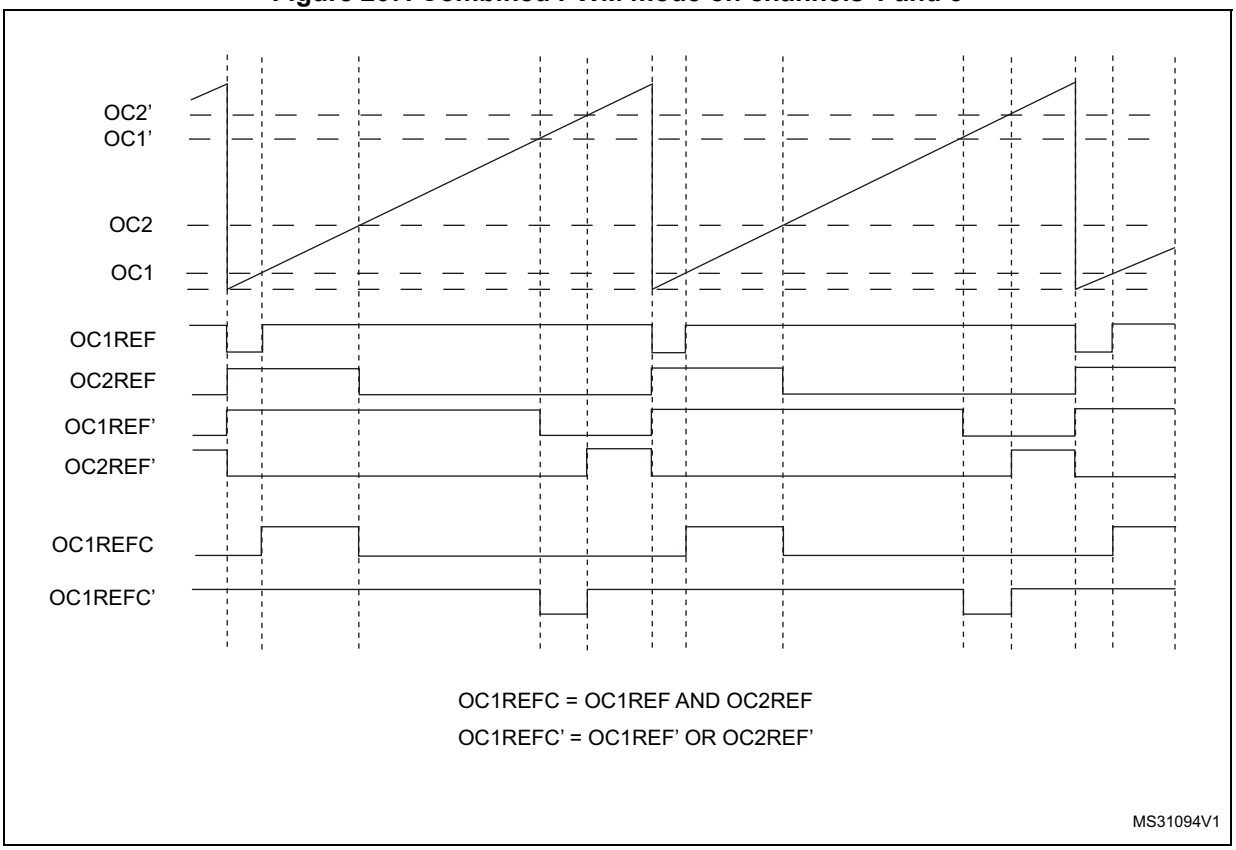
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 207 shows an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

Figure 207. Combined PWM mode on channels 1 and 3



24.3.12 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

OCREF_CLR_INPUT can be selected between the OCREF_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next update event (UEV) occurs.

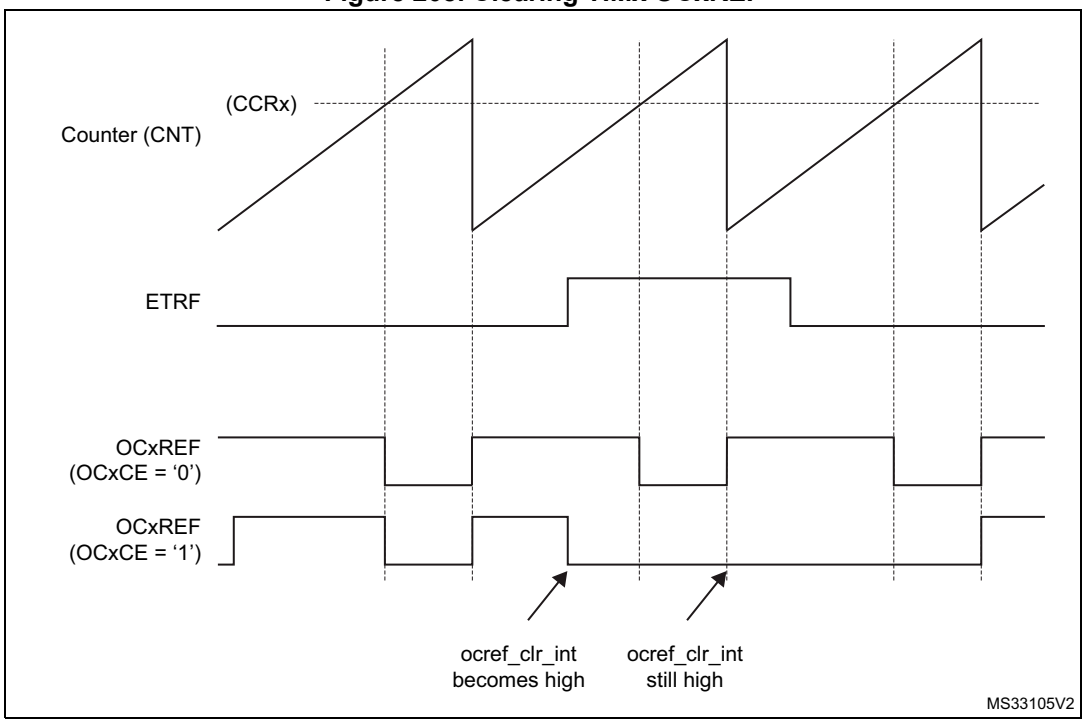
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 208 shows the behavior of the OCxREF signal when the ETRF input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 208. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

24.3.13 One-pulse mode

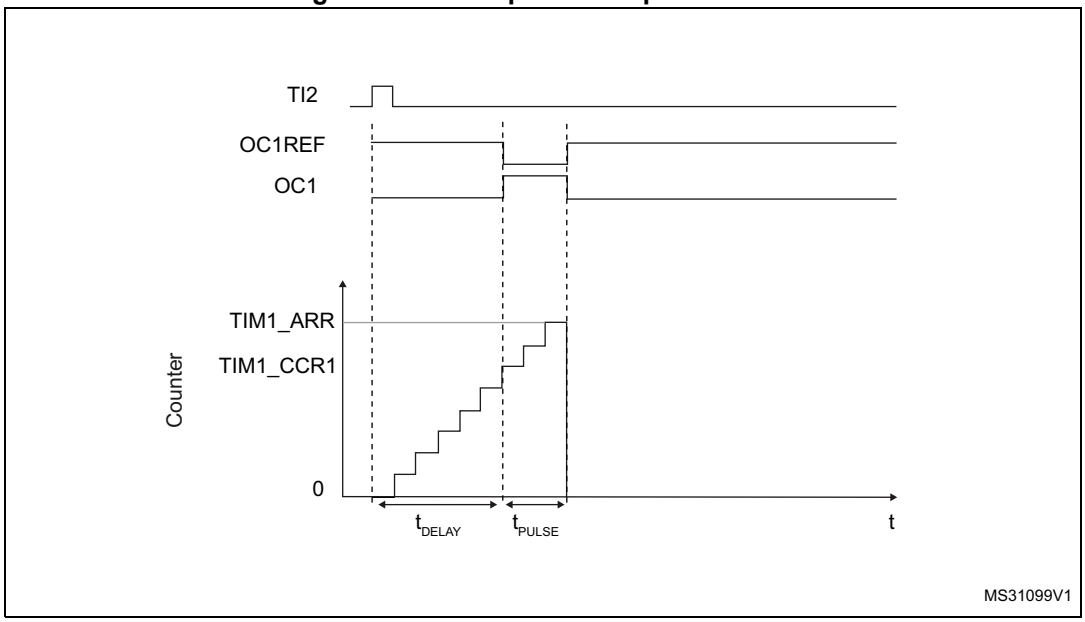
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT<CCRx ≤ ARR (in particular, 0<CCRx),

Figure 209. Example of one-pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

24.3.14 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 24.3.13*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

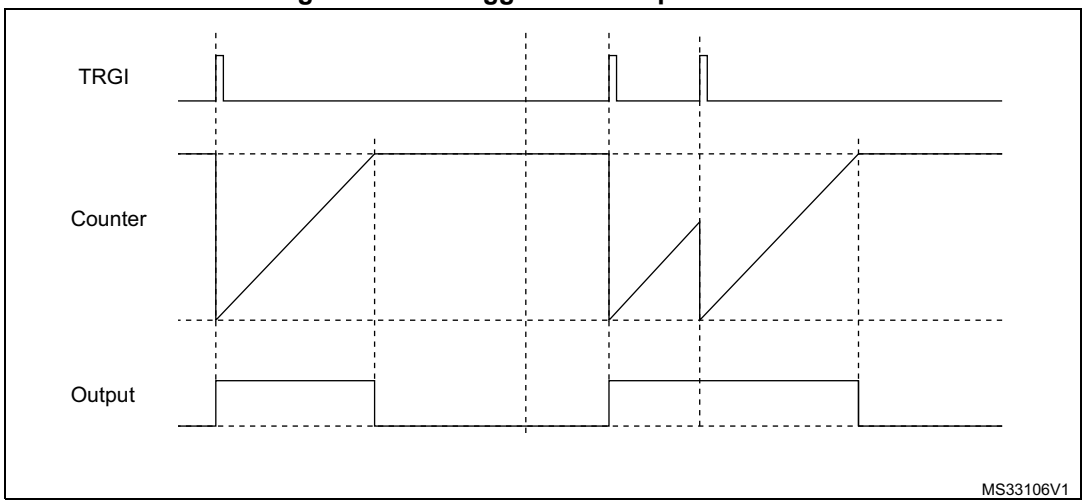
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

Note: In retriggerable one pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 210. Retriggerable one-pulse mode.



24.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 170*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 170. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Figure 211 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = '0' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = '0' (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

Figure 211. Example of counter operation in encoder interface mode

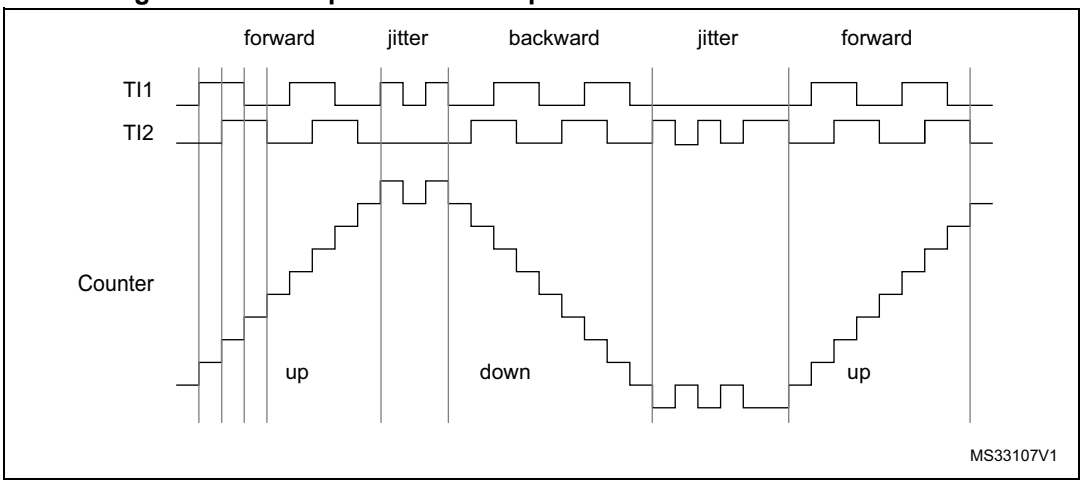
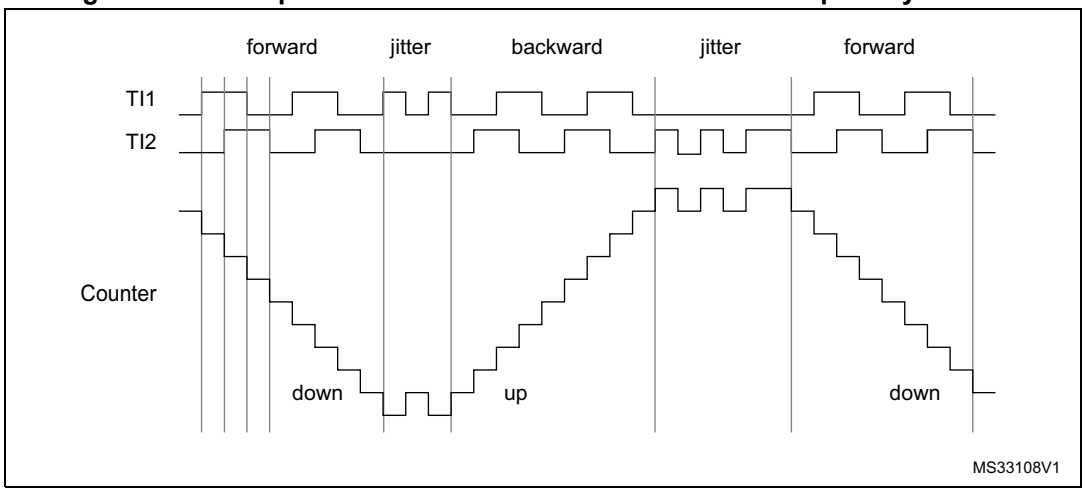


Figure 212 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 212. Example of encoder interface mode with TI1FP1 polarity inverted



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

24.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This permits to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

24.3.17 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 23.3.25: Interfacing with Hall sensors on page 670*.

24.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

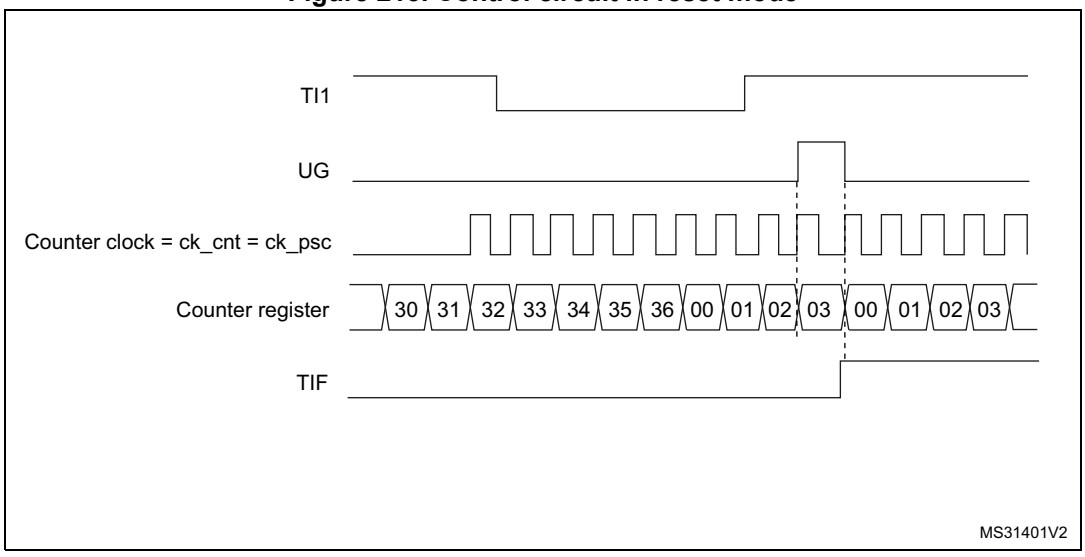
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 213. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

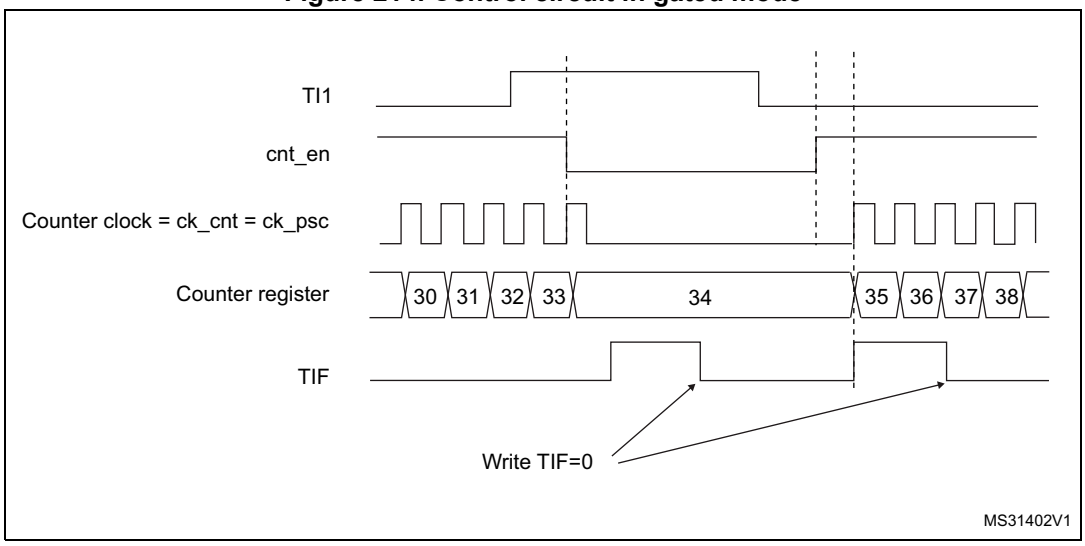
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 214. Control circuit in gated mode



1. The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Note: The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write

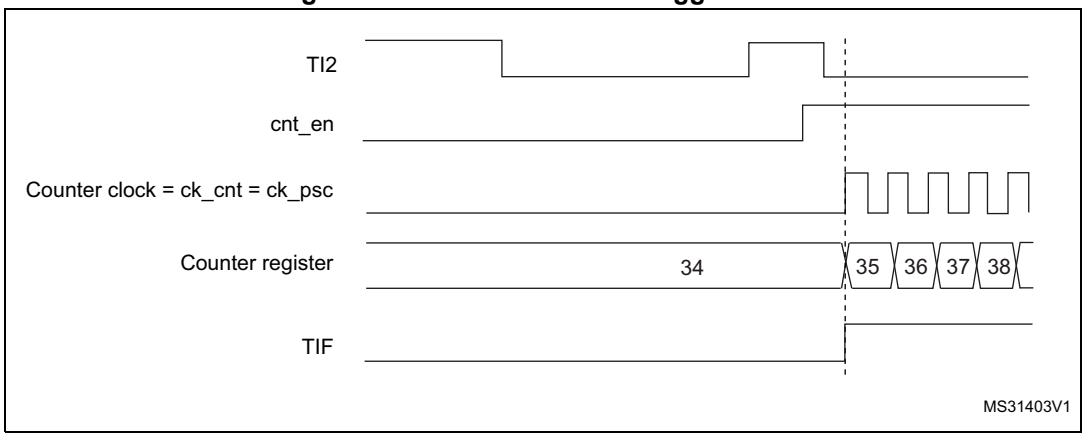
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 215. Control circuit in trigger mode



Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

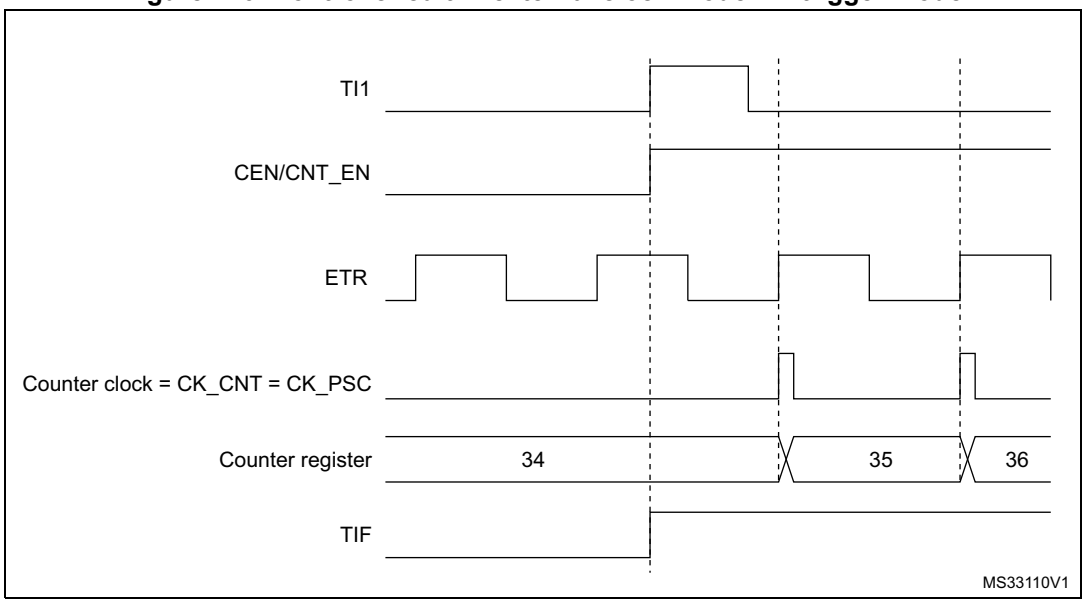
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 216. Control circuit in external clock mode 2 + trigger mode



24.3.19 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 217: Master/Slave timer example and *Figure 218: Master/slave connection example with 1 channel only timers* present an overview of the trigger selection and the master mode selection blocks.

Figure 217. Master/Slave timer example

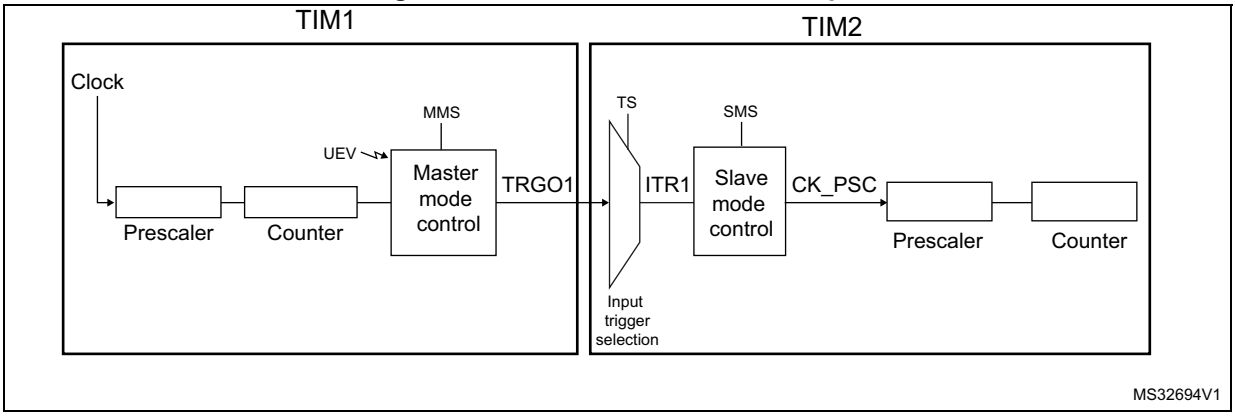
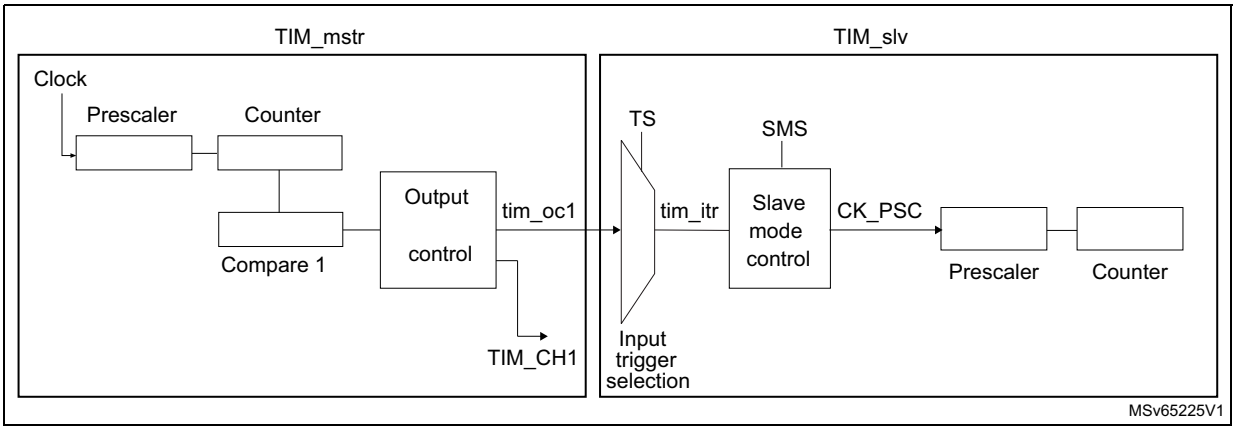


Figure 218. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 218) do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the "TIMx internal trigger connection" table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.
The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger.
For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

Using one timer as prescaler for another timer

For example, TIM1 can be configured to act as a prescaler for TIM2. Refer to *Figure 217*. To do this:

1. Configure TIM1 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM1_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM1 to TIM2, TIM2 must be configured in slave mode using ITR0 as internal trigger. This is selected through the TS bits in the TIM2_SMCR register (writing TS=00000).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM1 trigger signal (which correspond to the TIM1 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM1 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

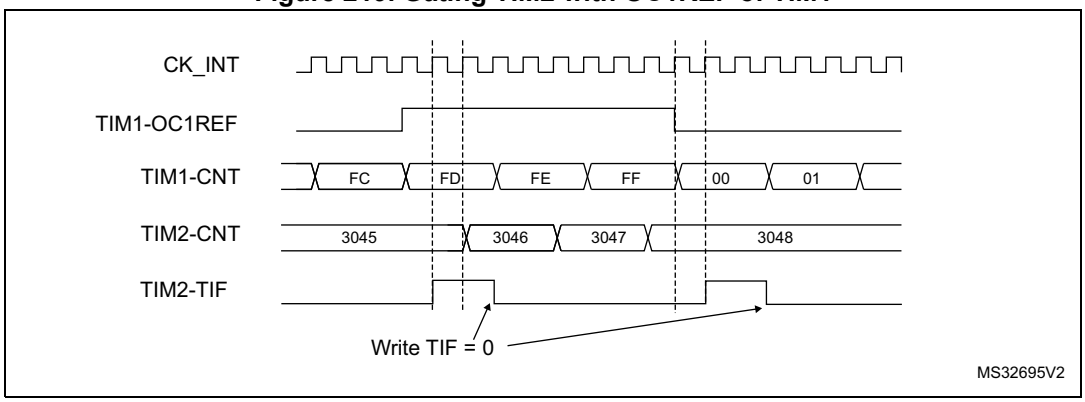
Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 1. Refer to *Figure 217* for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM1 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

1. Configure TIM1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
2. Configure the TIM1 OC1REF waveform (TIM1_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Enable TIM2 by writing ‘1 in the CEN bit (TIM_CR1 register).
6. Start TIM by writing ‘1 in the CEN bit (TIM1_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

Figure 219. Gating TIM2 with OC1REF of TIM1

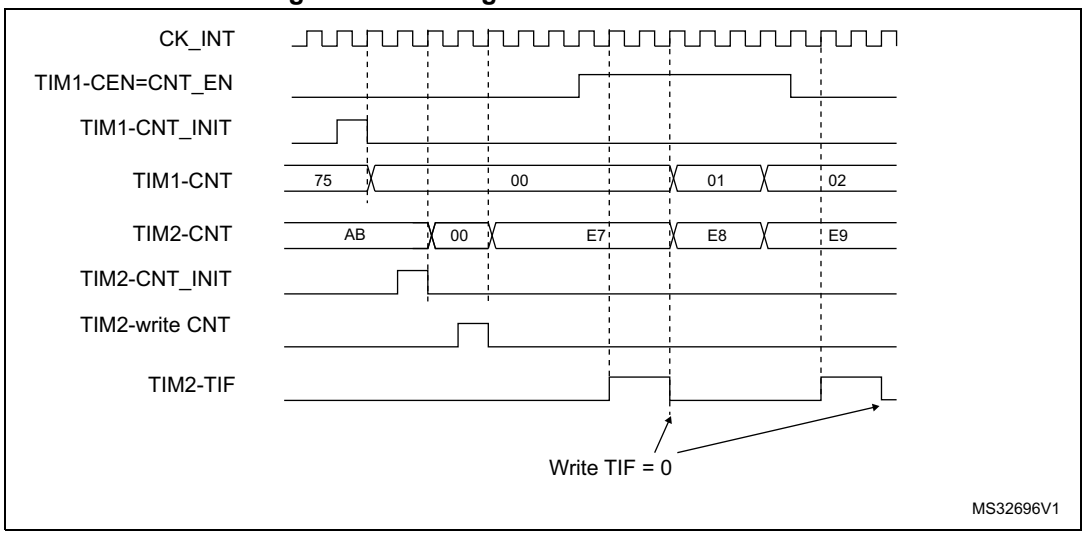


In the example in *Figure 219*, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM1. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 220*), we synchronize TIM1 and TIM2. TIM1 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM1 is disabled by writing ‘0 to the CEN bit in the TIM1_CR1 register:

1. Configure TIM1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
2. Configure the TIM1 OC1REF waveform (TIM1_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM1 by writing ‘1 in UG bit (TIM1_EGR register).
6. Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
7. Initialize TIM2 to 0xE7 by writing ‘0xE7’ in the TIM2 counter (TIM2_CNTL).
8. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
9. Start TIM1 by writing ‘1 in the CEN bit (TIM1_CR1 register).
10. Stop TIM1 by writing ‘0 in the CEN bit (TIM1_CR1 register).

Figure 220. Gating TIM2 with Enable of TIM1

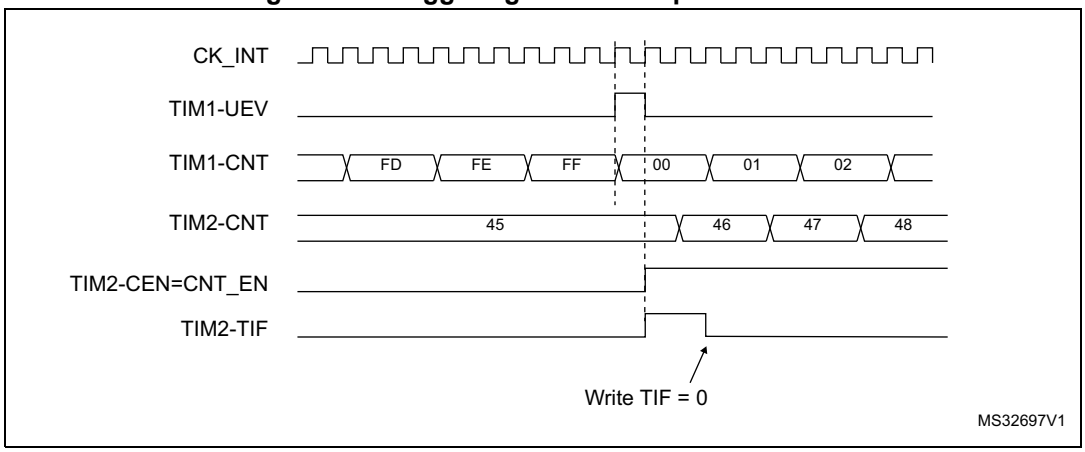


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to *Figure 217* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

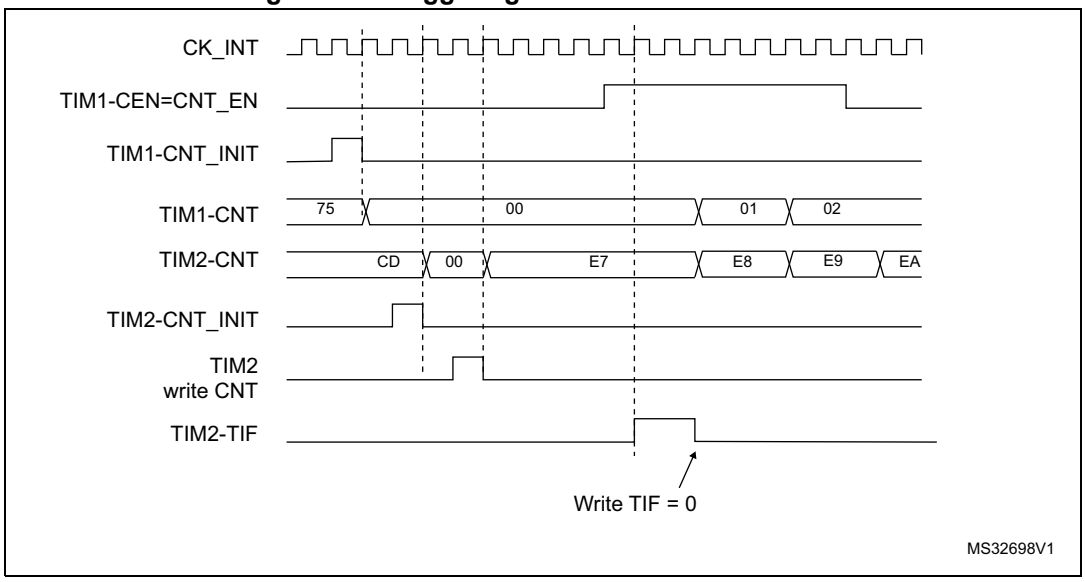
1. Configure TIM1 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM1_CR2 register).
2. Configure the TIM1 period (TIM1_ARR registers).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM1 by writing ‘1 in the CEN bit (TIM1_CR1 register).

Figure 221. Triggering TIM2 with update of TIM1



As in the previous example, both counters can be initialized before starting counting. *Figure 222* shows the behavior with the same configuration as in *Figure 221* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 222. Triggering TIM2 with Enable of TIM1



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

24.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

24.3.21 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIM2_STOP configuration bit in DBGMCU module. For more details, refer to *Section 36.11.3: DBGMCU APB1 peripheral freeze register 1 (DBGMCU_APB1FZR1)*.

24.4 TIM2 registers

In this section, “TIMx” should be understood as “TIM2” since there is only one instance of this type of timer for the products to which this reference manual applies.

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

24.4.1 TIM2 control register 1 (TIM2_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

24.4.2 TIM2 control register 2 (TIM2_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
See also *Section 23.3.25: Interfacing with Hall sensors on page 670*

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits permit to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)
100: **Compare** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

24.4.3 TIM2 slave mode control register (TIM2_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.
00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010: Internal Trigger 2 (ITR2)
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
00110: Filtered Timer Input 2 (TI2FP2)
00111: External Trigger input (ETRF)
01000: Internal Trigger 4 (ITR4)
01001: Internal Trigger 5 (ITR5)
01010: Internal Trigger 6 (ITR6)
01011: Internal Trigger 7 (ITR7)
01100: Internal Trigger 8 (ITR8)
Others: Reserved
See *Table 171: TIM2 internal trigger connection on page 769* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source
0: OCREF_CLR_INT is connected to the OCREF_CLR input
1: OCREF_CLR_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=00100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 171. TIM2 internal trigger connection

Slave TIM	ITR0	ITR1	ITR2 - ITR8
TIM2	TIM1	-	-

24.4.4 TIM2 DMA/Interrupt enable register (TIM2_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
			rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable

0: CC4 DMA request disabled.

1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

24.4.5 TIM2 status register (TIM2_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS=0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

24.4.6 TIM2 event generation register (TIM2_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

24.4.7 TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

24.4.8 TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

24.4.9 TIM2 capture/compare mode register 2 [alternate] (TIM2_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

24.4.10 TIM2 capture/compare mode register 2 [alternate] (TIM2_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

24.4.11 TIM2 capture/compare enable register (TIM2_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*
CC1 channel configured as output: CC1NP must be kept cleared in this case.
CC1 channel configured as input: This bit is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*
0: Capture mode disabled / OC1 is not active
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 172. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO control and alternate function registers.

24.4.12 TIM2 counter [alternate] (TIM2_CNT)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- This section is for UIFREMAP = 0
- Next section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNT[31:0]**: counter value

24.4.13 TIM2 counter [alternate] (TIM2_CNT)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- Previous section is for UIFREMAP = 0
- This section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	CNT[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:0 **CNT[30:0]**: counter value

24.4.14 TIM2 prescaler (TIM2_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

24.4.15 TIM2 auto-reload register (TIM2_ARR)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARR[31:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 24.3.1: Time-base unit on page 721* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

24.4.16 TIM2 capture/compare register 1 (TIM2_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

24.4.17 TIM2 capture/compare register 2 (TIM2_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

24.4.18 TIM2 capture/compare register 3 (TIM2_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/Compare value

If channel CC3 is configured as output:
CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.
If channel CC3is configured as input:
CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

24.4.19 TIM2 capture/compare register 4 (TIM2_CCR4)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/Compare value

1. if CC4 channel is configured as output (CC4S bits):
   CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
   The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.
2. if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):
   CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

24.4.20 TIM2 DMA control register (TIM2_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...
Example: Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

24.4.21 TIM2 DMA address for full transfer (TIM2_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

24.4.22 TIM2 option register 1 (TIM2_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI4_RMP[1:0]		ETR_RMP	Res.
												rw	rw	rw	

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **TI4_RMP[1:0]**: Timer input 4 remap

Set and cleared by software.

00: TIM2 TI4 is connected to GPIO: Refer to Alternate Function mapping

01: TIM2 TI4 is connected to COMP1_OUT

10: TIM2 TI4 is connected to COMP2_OUT

11: TIM2 TI4 is connected to a logical OR between COMP1_OUT and COMP2_OUT

Bit 1 **ETR_RMP**: External trigger 1 remap

Set and cleared by software.

0: TIM2 ETR is connected to GPIO: Refer to Alternate Function mapping

1: LSE internal clock is connected to TIM2_ETR input

Bit 0 Reserved, must be kept at reset value.

24.4.23 TIM2 alternate function option register 1 (TIM2_AF1)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection
These bits select the ETR input source.
0000: GPIO or LSE internal clock, as per ETR_RMP bit in TIM2_OR1
0001: COMP1
0010: COMP2
Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

24.4.24 TIM2 timer input selection register (TIM2_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM2_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM2_CH1 input
Others: Reserved

24.4.25 TIMx register map

TIMx registers are mapped as described in the table below:

Table 173. TIM2 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																				0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0

Table 173. TIM2 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**TIMx_CNT**	CNT[31] or UIFCPY	CNT[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	Reserved																																
0x34	**TIMx_CCR1**	CCR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	CCR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIMx_CCR3**	CCR3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIMx_CCR4**	CCR4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved																																
0x48	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM2_OR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI4_RMP		ETR_RMP	Res.
	Reset value																													0	0	0	
0x60	**TIM2_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value															0	0	0	0														

Table 173. TIM2 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x68	**TIM2_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0

Refer to *Section 2.4 on page 62* for the register boundary addresses.

25 General-purpose timers (TIM16/TIM17)

25.1 TIM16/TIM17 introduction

The TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

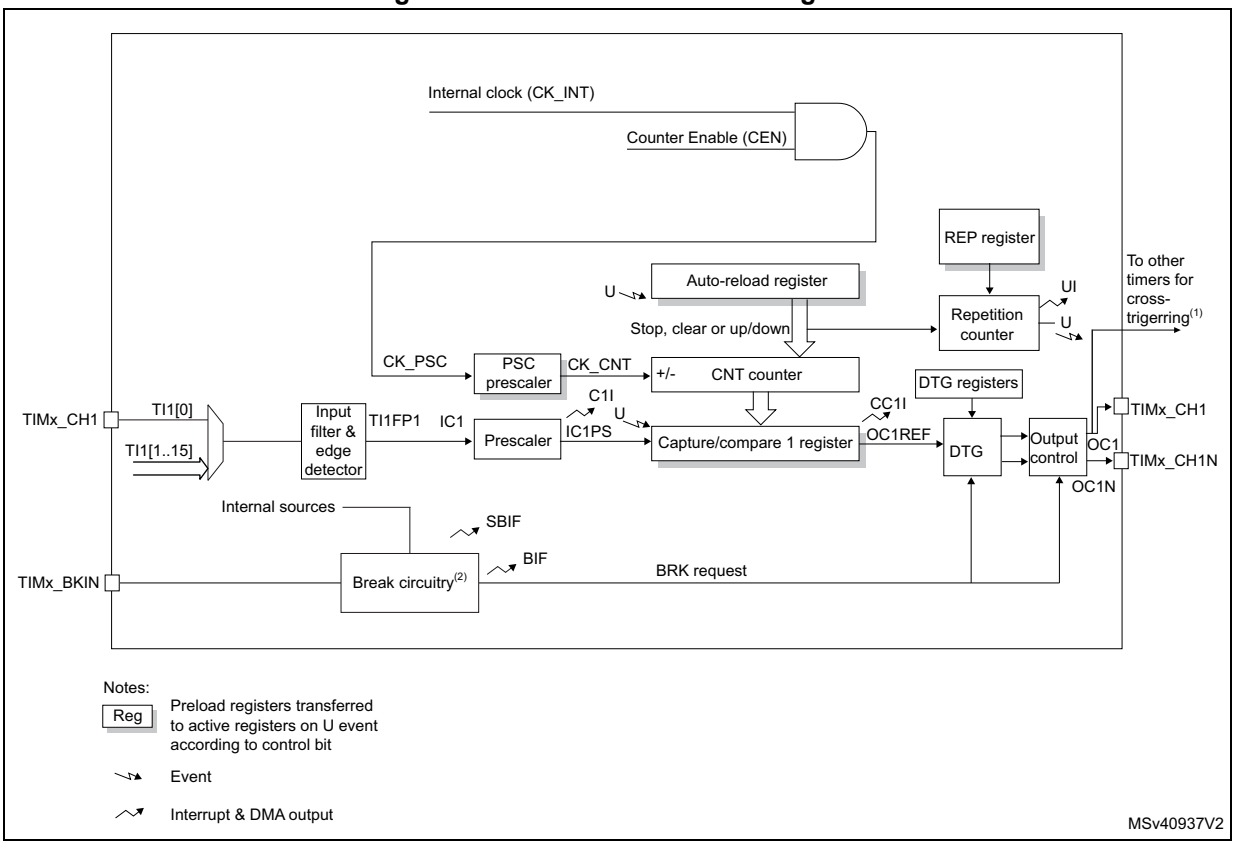
The TIM16/TIM17 timers are completely independent, and do not share any resources.

25.2 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

Figure 223. TIM16/TIM17 block diagram



1. This signal can be used as trigger for some slave timer, see *Section 25.3.18: Using timer output as trigger for other timers (TIM16/TIM17)*.

25.3 TIM16/TIM17 functional description

25.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 224 and *Figure 225* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 224. Counter timing diagram with prescaler division change from 1 to 2

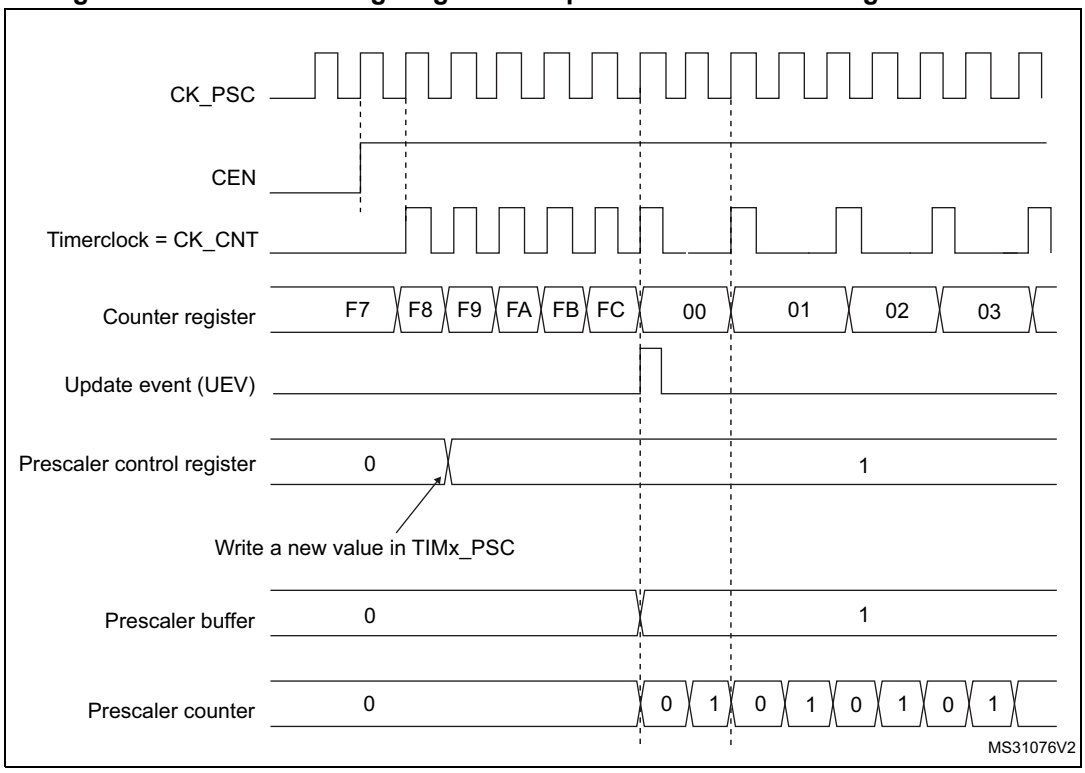
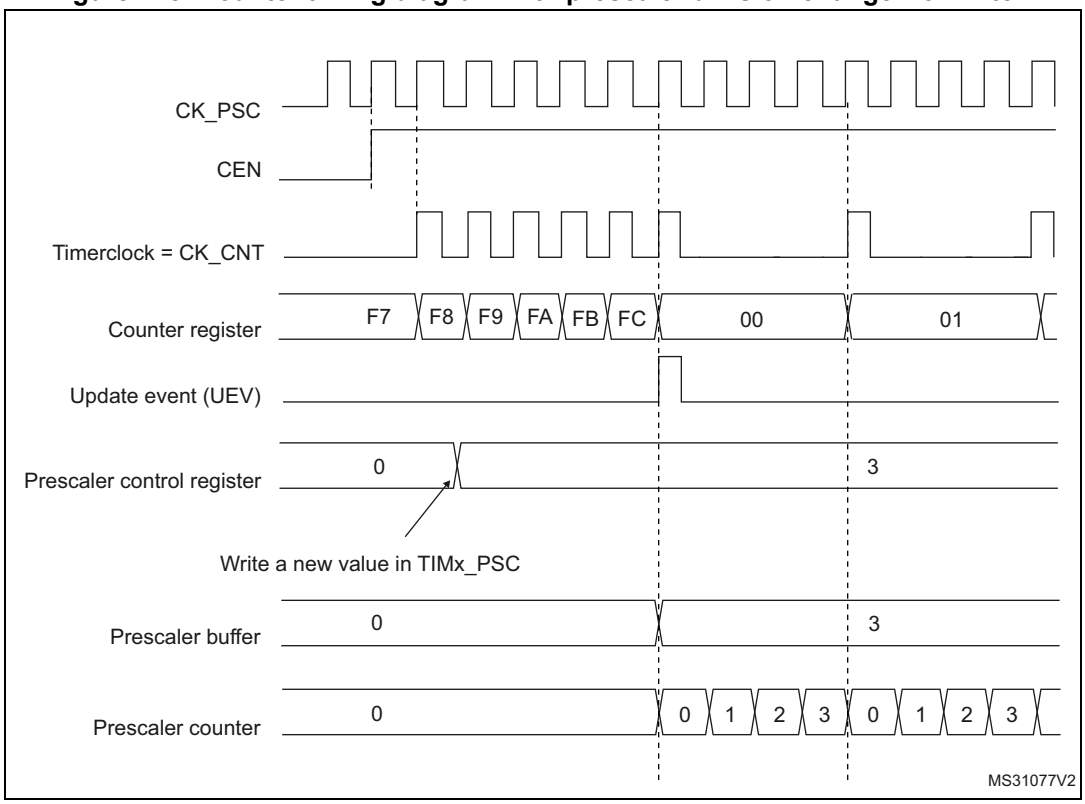


Figure 225. Counter timing diagram with prescaler division change from 1 to 4



25.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 226. Counter timing diagram, internal clock divided by 1

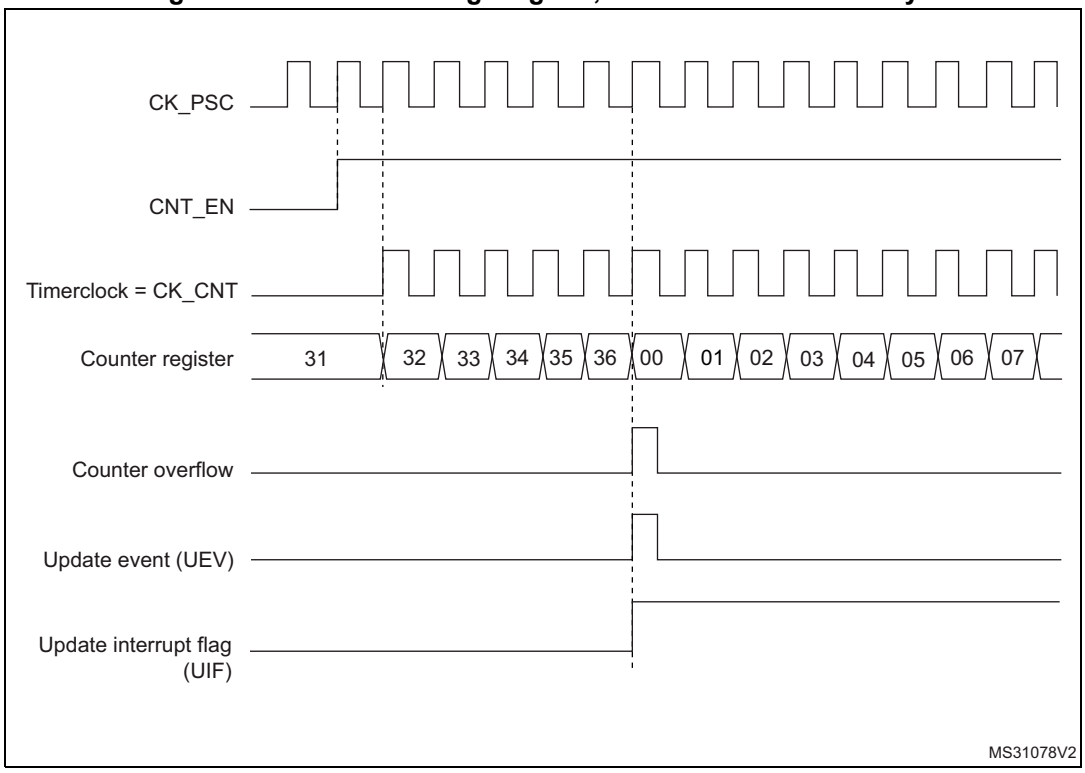


Figure 227. Counter timing diagram, internal clock divided by 2

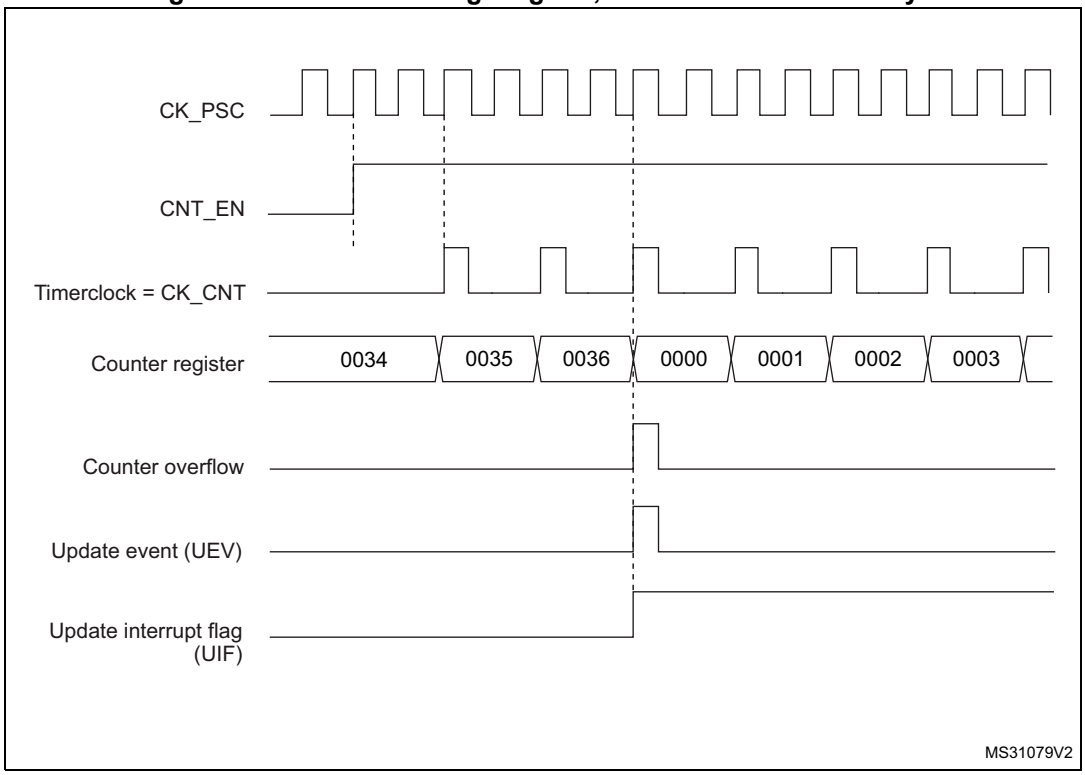


Figure 228. Counter timing diagram, internal clock divided by 4

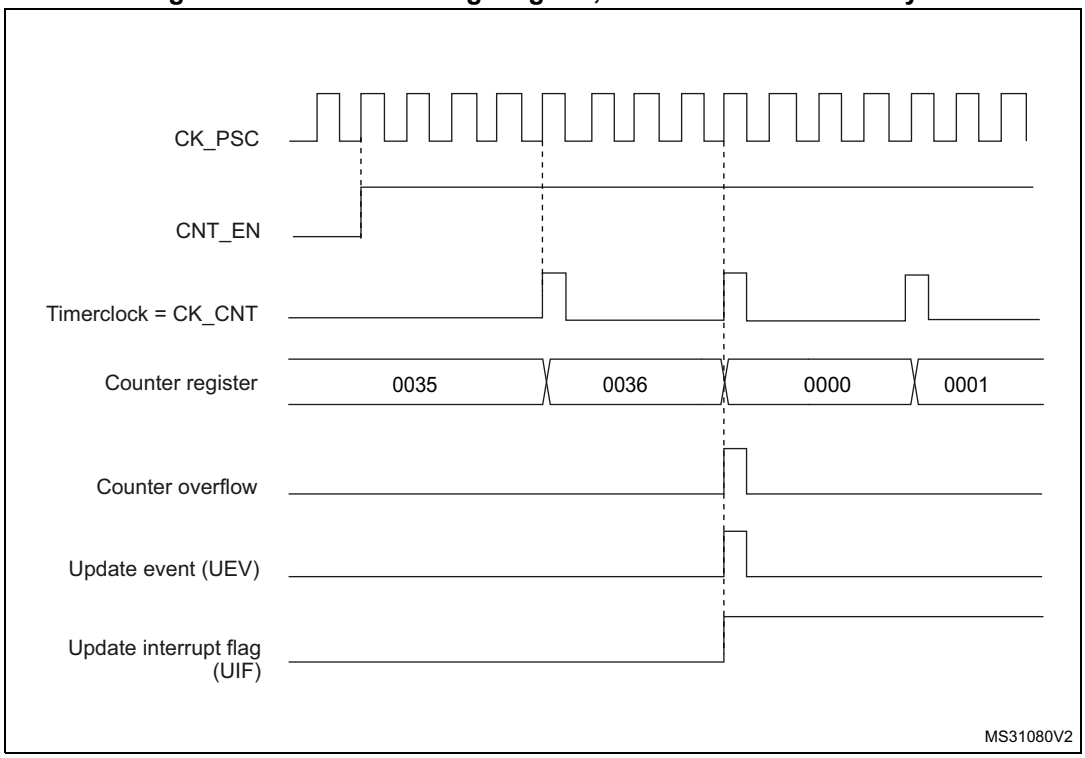


Figure 229. Counter timing diagram, internal clock divided by N

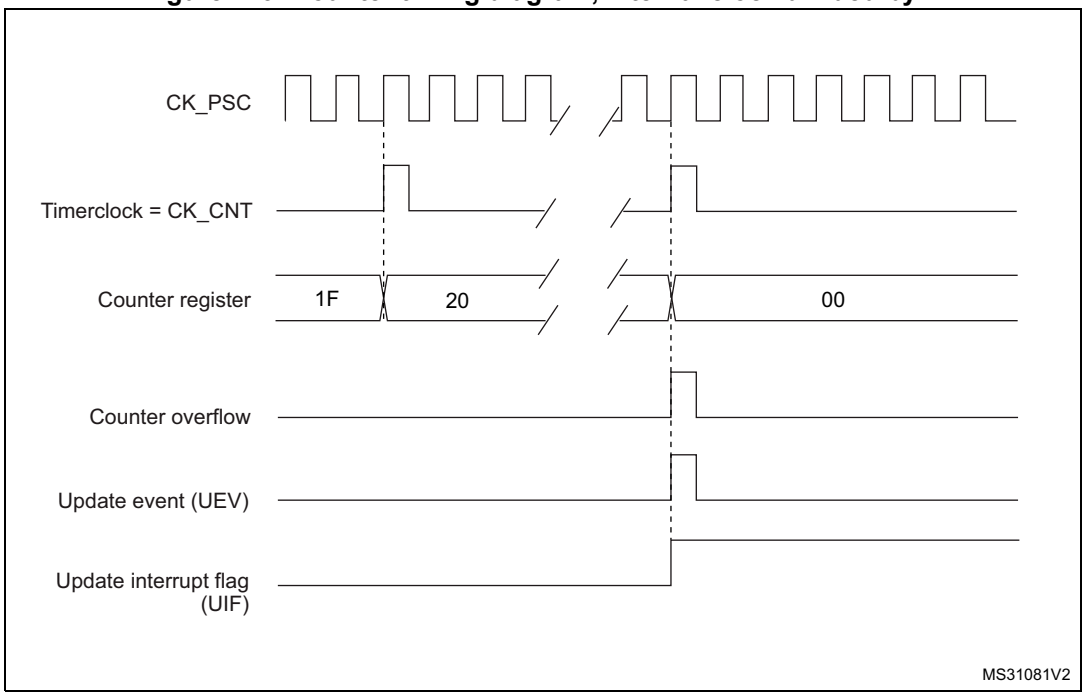


Figure 230. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

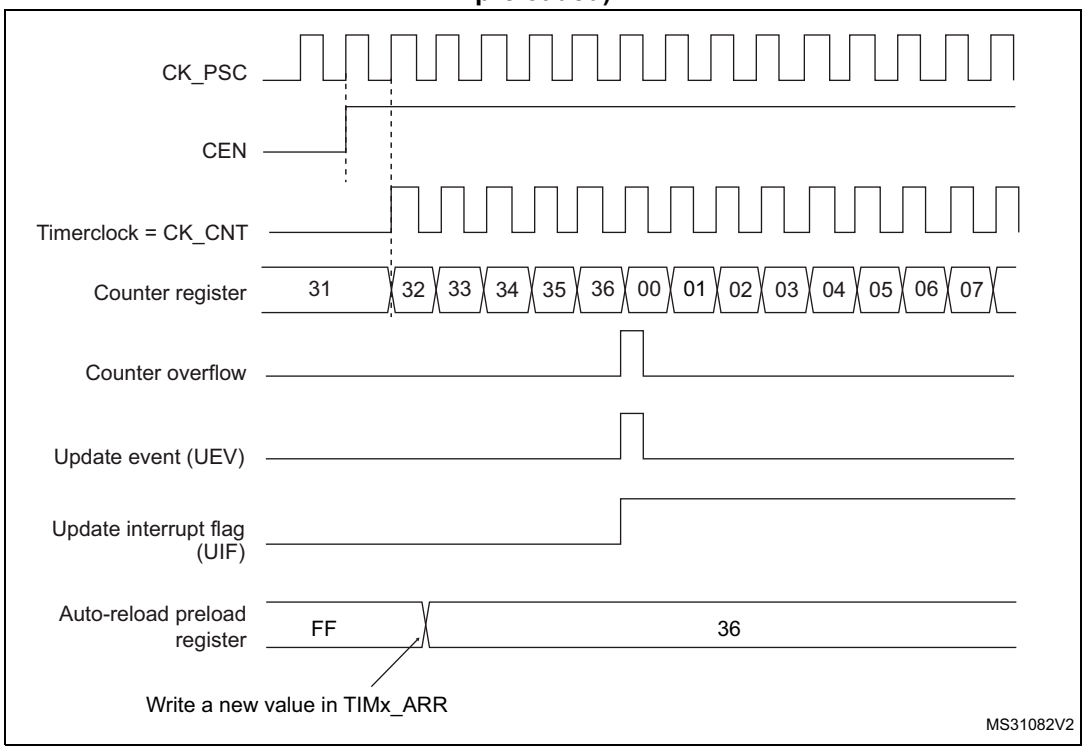
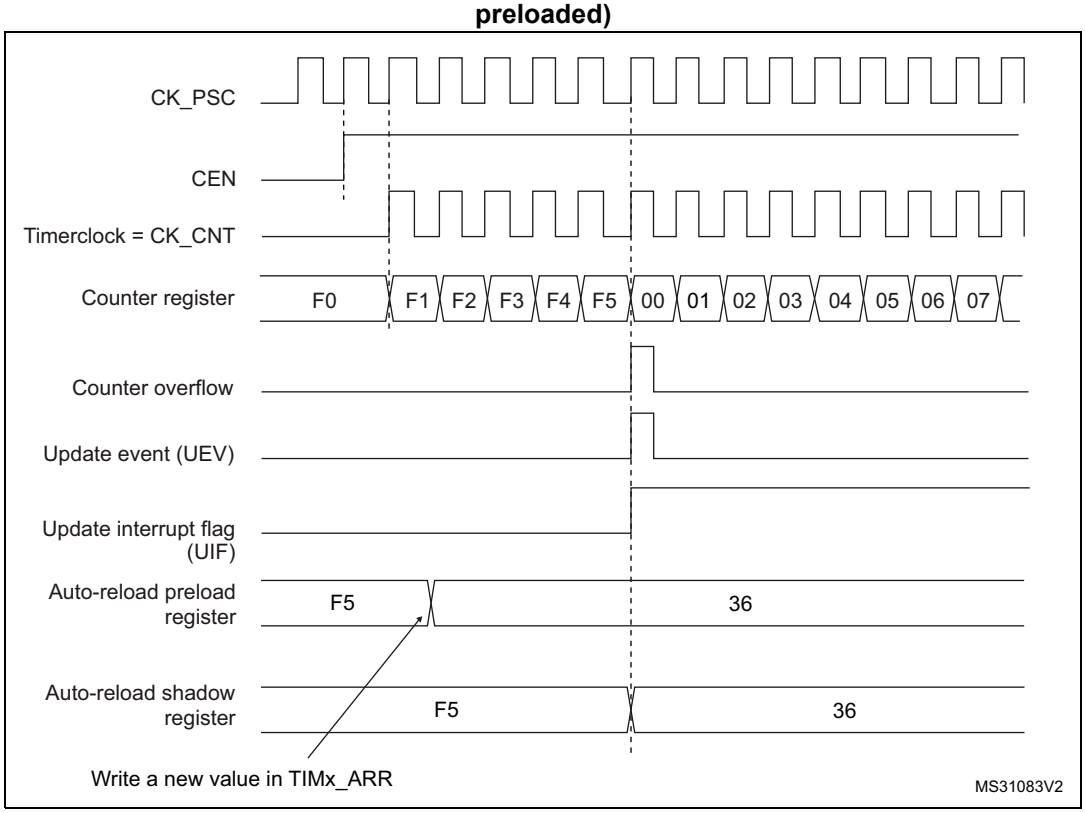


Figure 231. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



25.3.3 Repetition counter

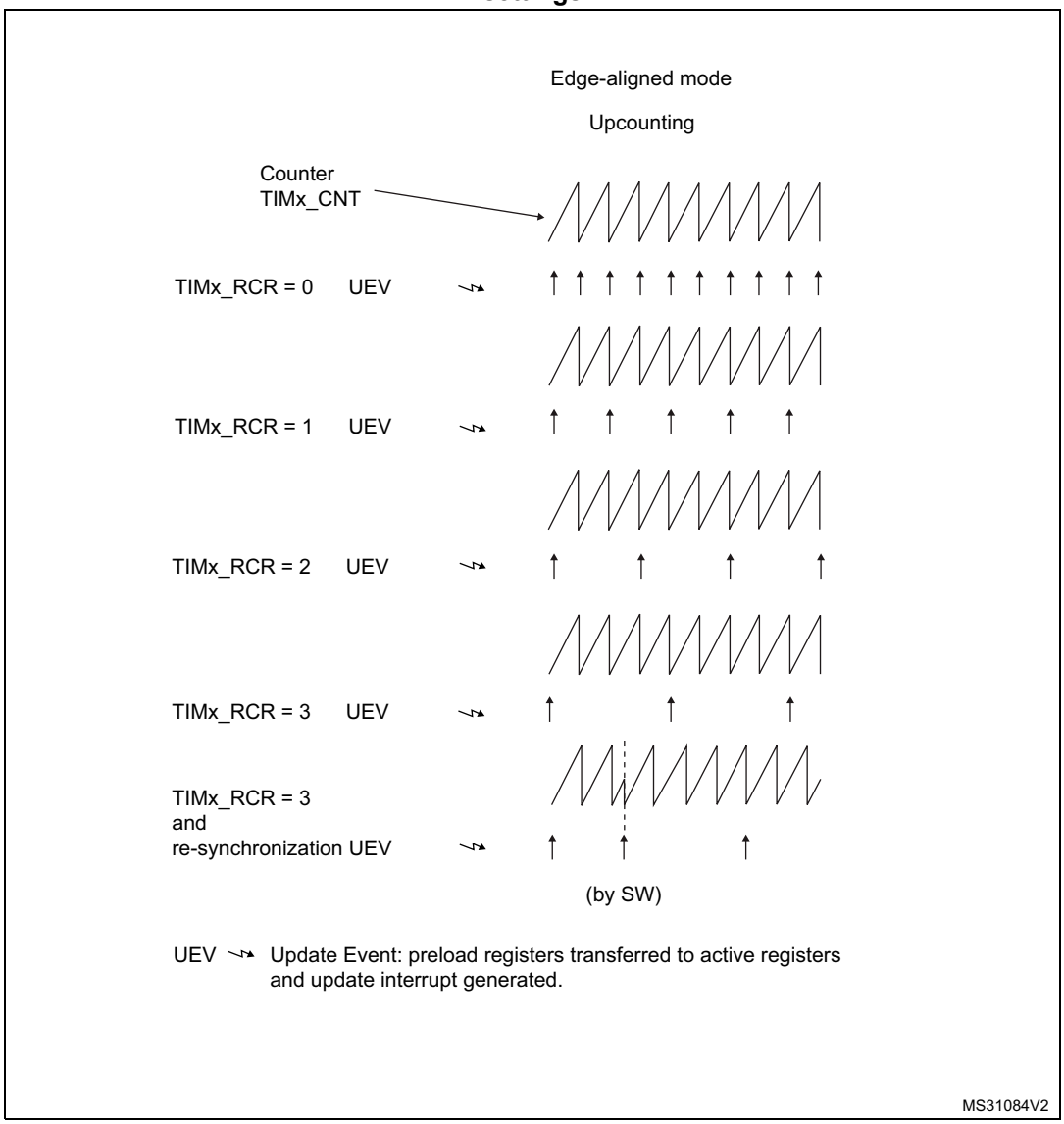
Section 25.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 232*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 232. Update rate examples depending on mode and TIMx_RCR register settings



25.3.4 Clock selection

The counter clock can be provided by the following clock sources:

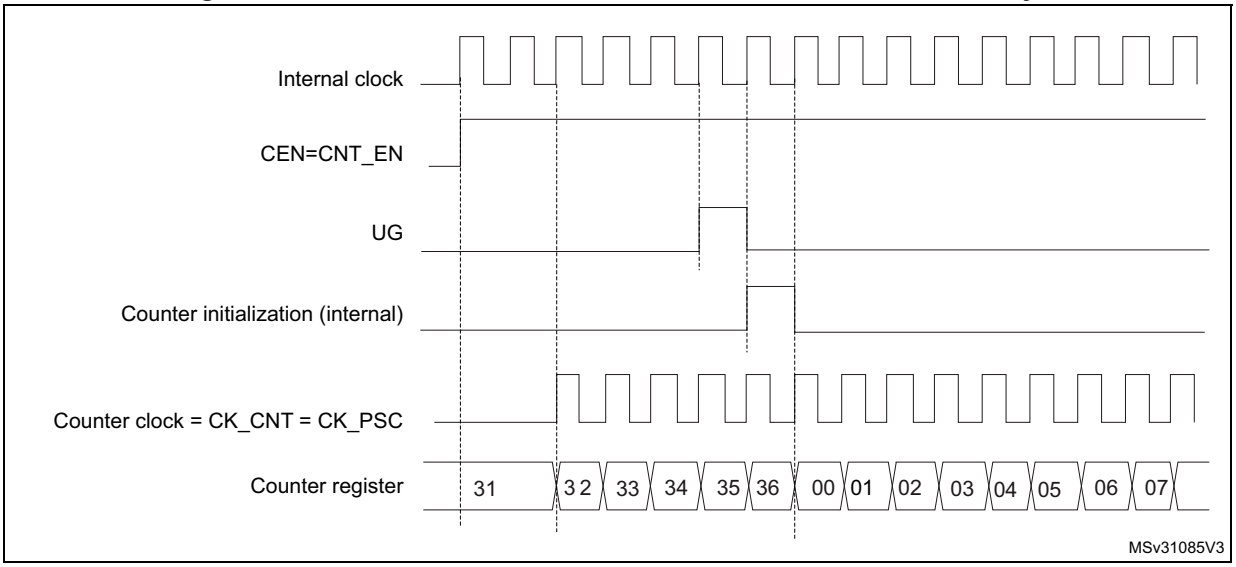
- Internal clock (CK_INT)
- External clock mode1: external input pin

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 233 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

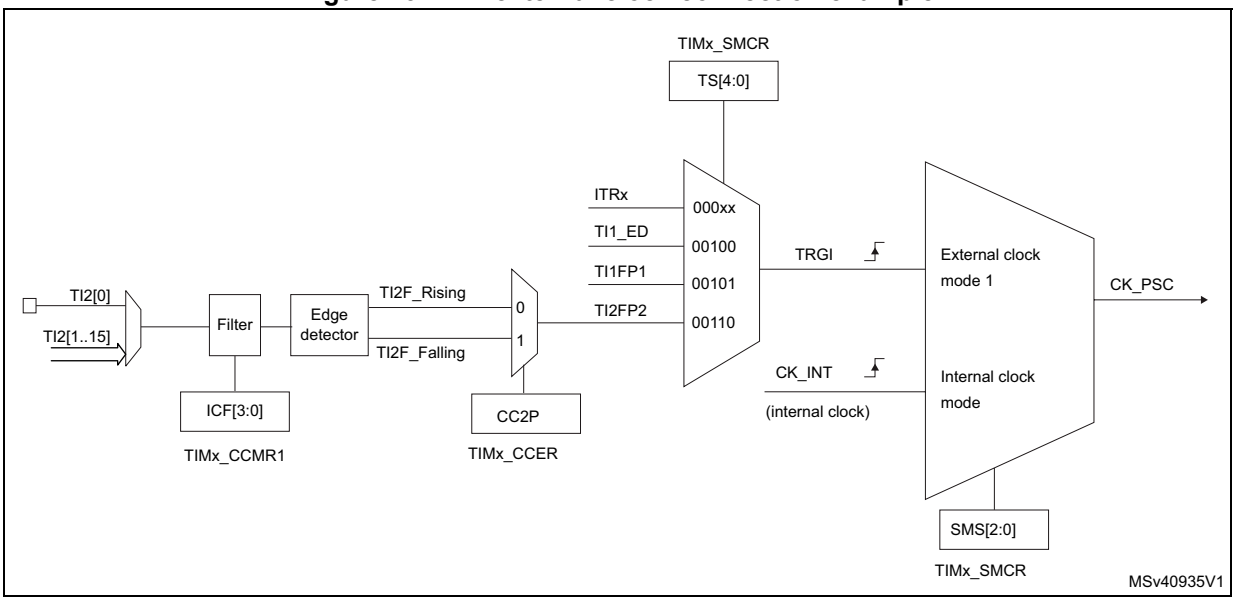
Figure 233. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 234. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

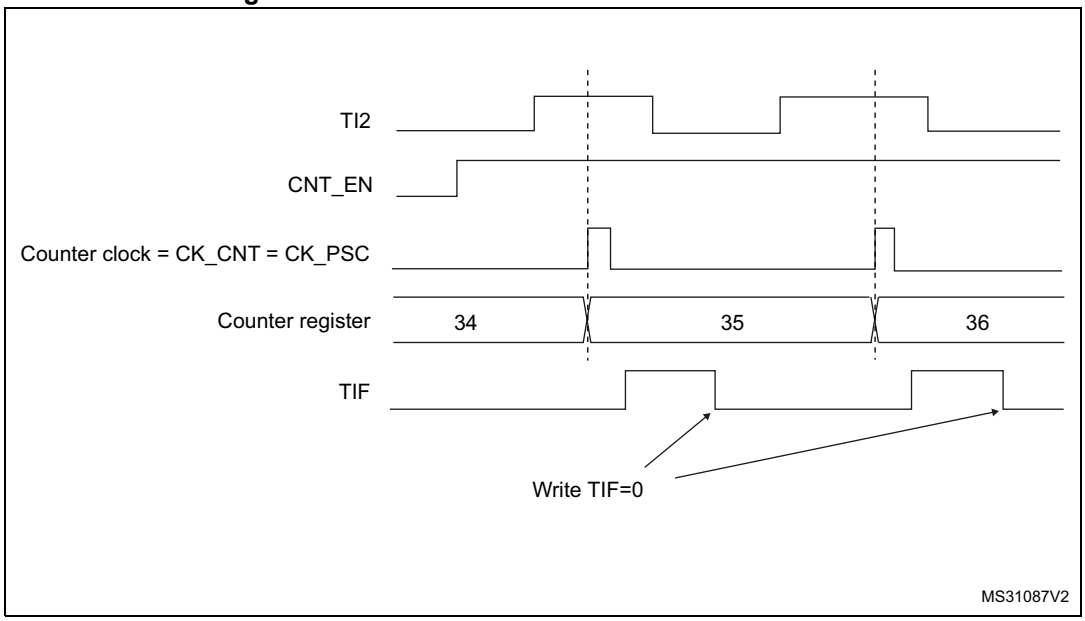
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so it does not need to be configured.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 235. Control circuit in external clock mode 1



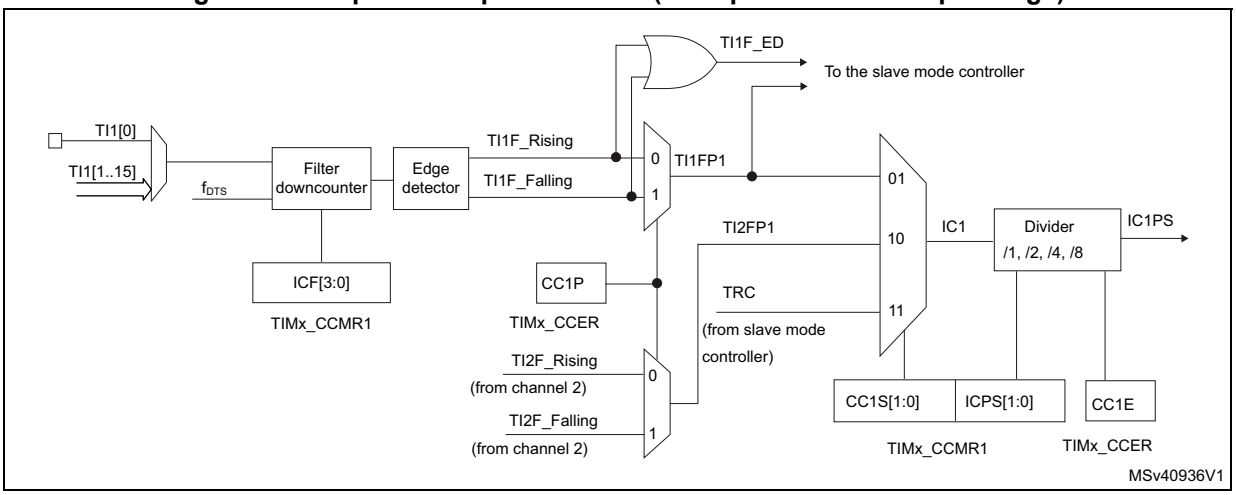
25.3.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 236 to *Figure 238* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 236. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 237. Capture/compare channel 1 main circuit

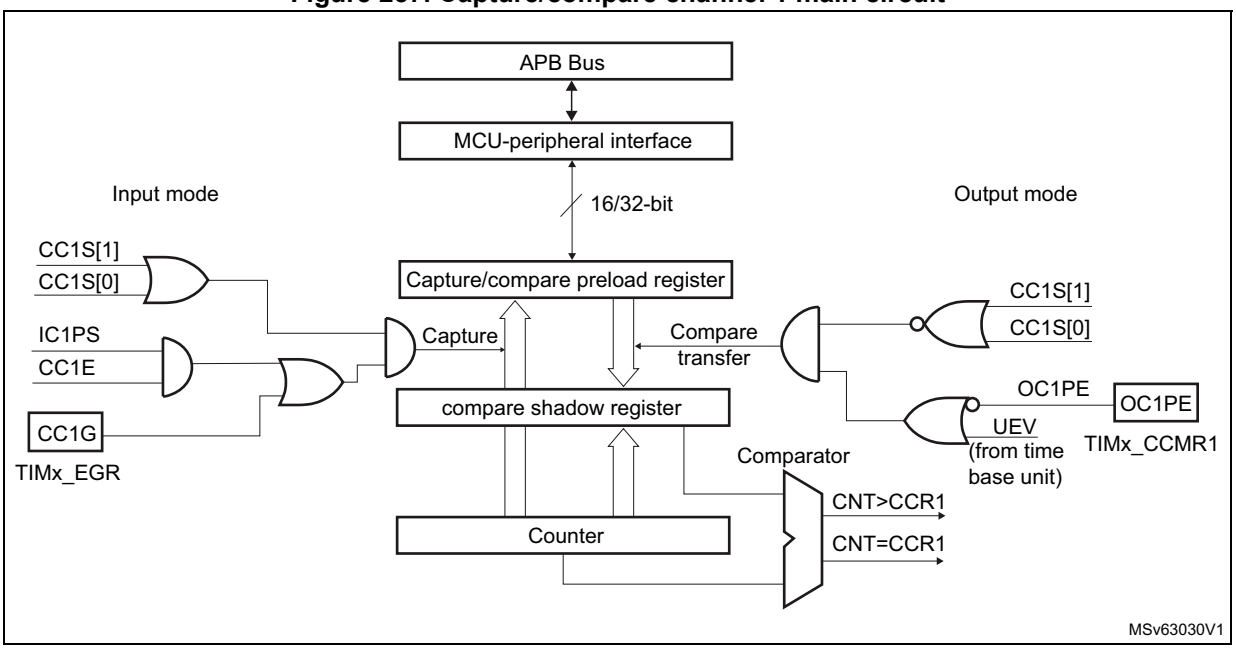
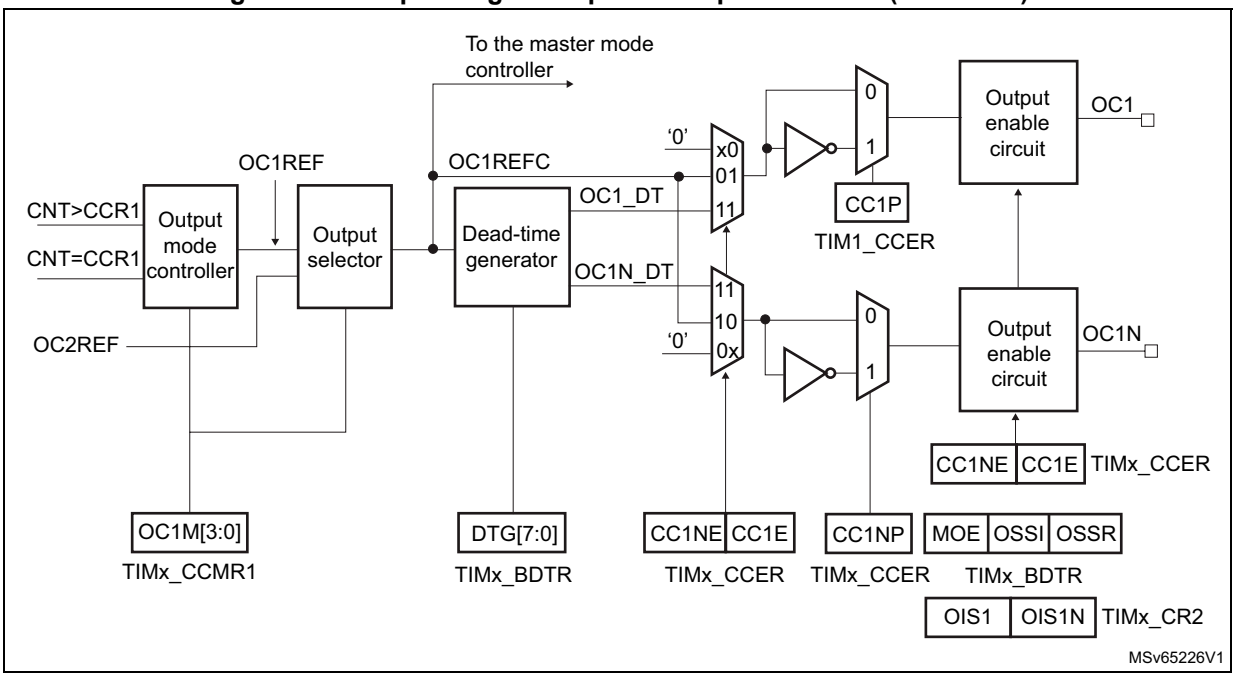


Figure 238. Output stage of capture/compare channel (channel 1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

25.3.6 Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

25.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

25.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

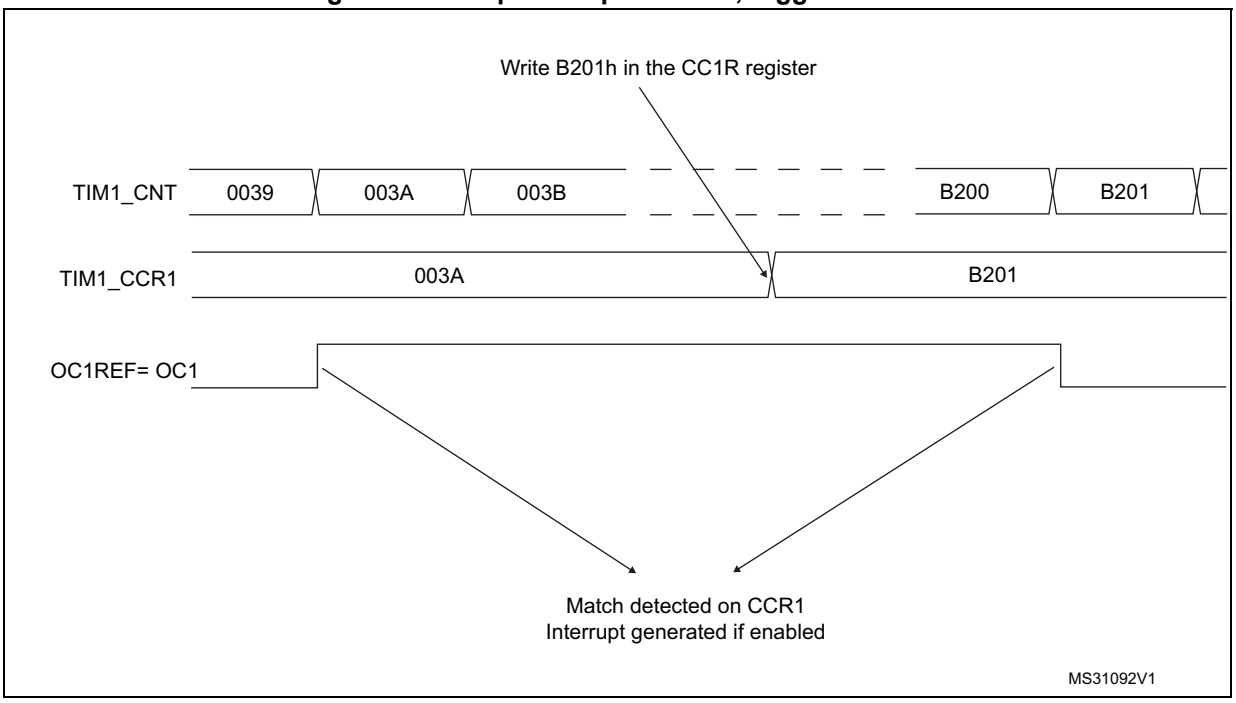
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 239*.

Figure 239. Output compare mode, toggle on OC1



25.3.9 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

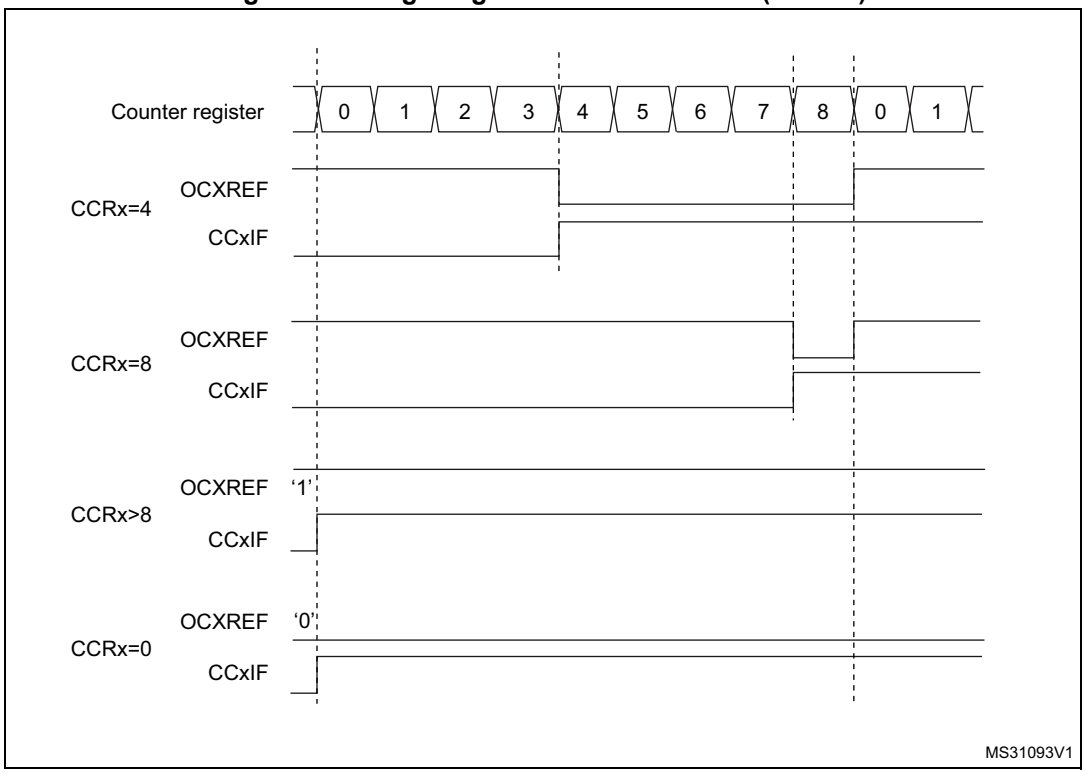
OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode on page 794*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. *Figure 240* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 240. Edge-aligned PWM waveforms (ARR=8)



25.3.10 Complementary outputs and dead-time insertion

The TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 175: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) on page 829* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 241. Complementary output with dead-time insertion.

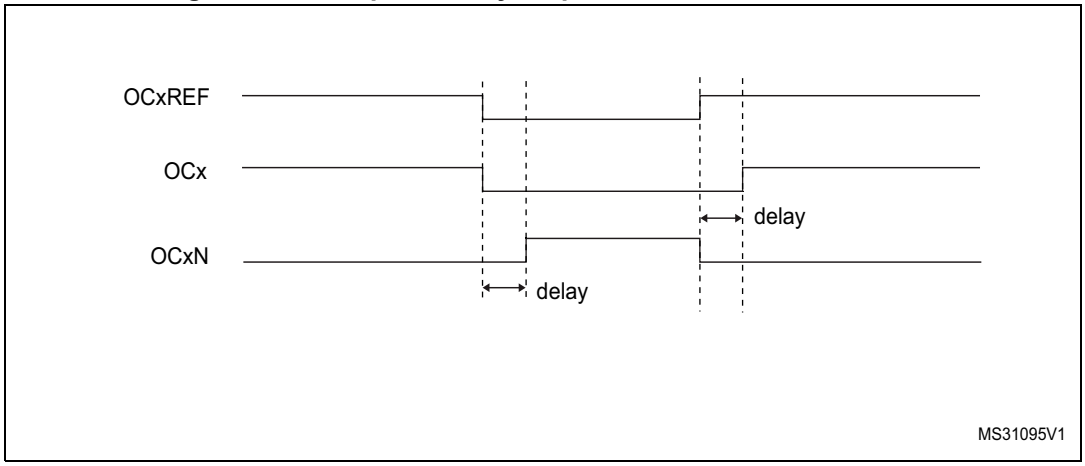


Figure 242. Dead-time waveforms with delay greater than the negative pulse.

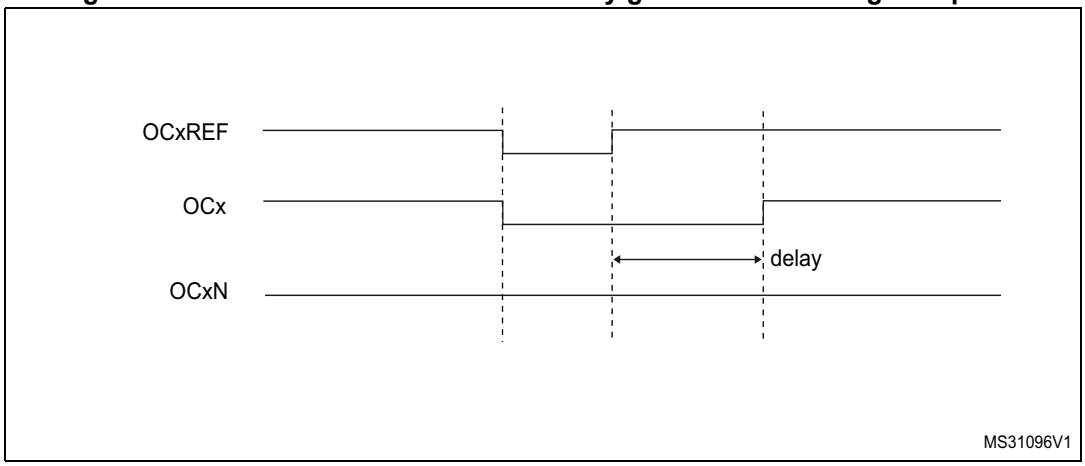
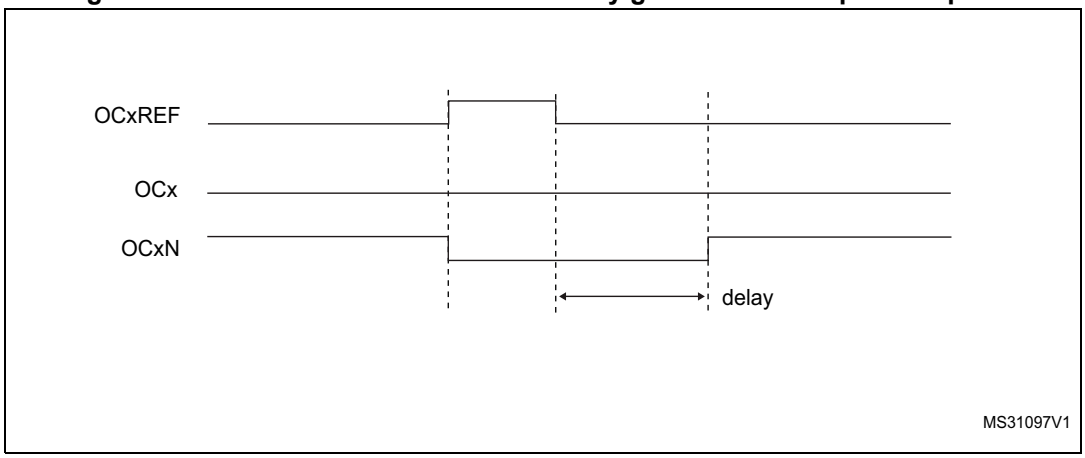


Figure 243. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 25.4.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 832* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

25.3.11 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay

must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs will be in idle state and forced to inactive level or Hi-Z depending on OSSI value.
If both the MOE and AOE bits are reset by the CPU, the outputs will be in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

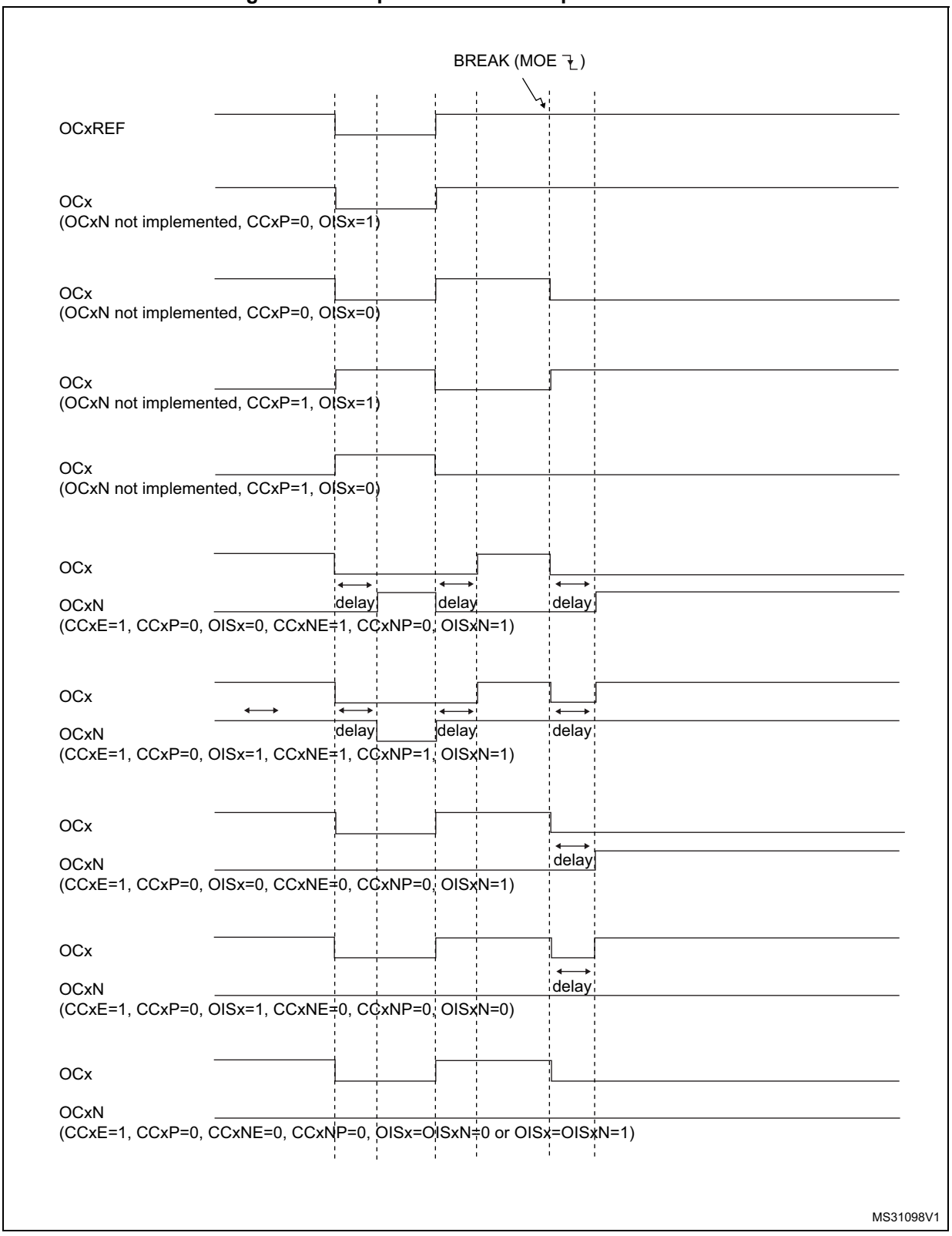
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 25.4.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 832*. The LOCK bits can be written only once after an MCU reset.

The *Figure 244* shows an example of behavior of the outputs in response to a break.

Figure 244. Output behavior in response to a break



25.3.12 Bidirectional break inputs

The TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 245*.

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (BG) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 174*)

Table 174. Break protection disarming conditions

MOE	BKDIR	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

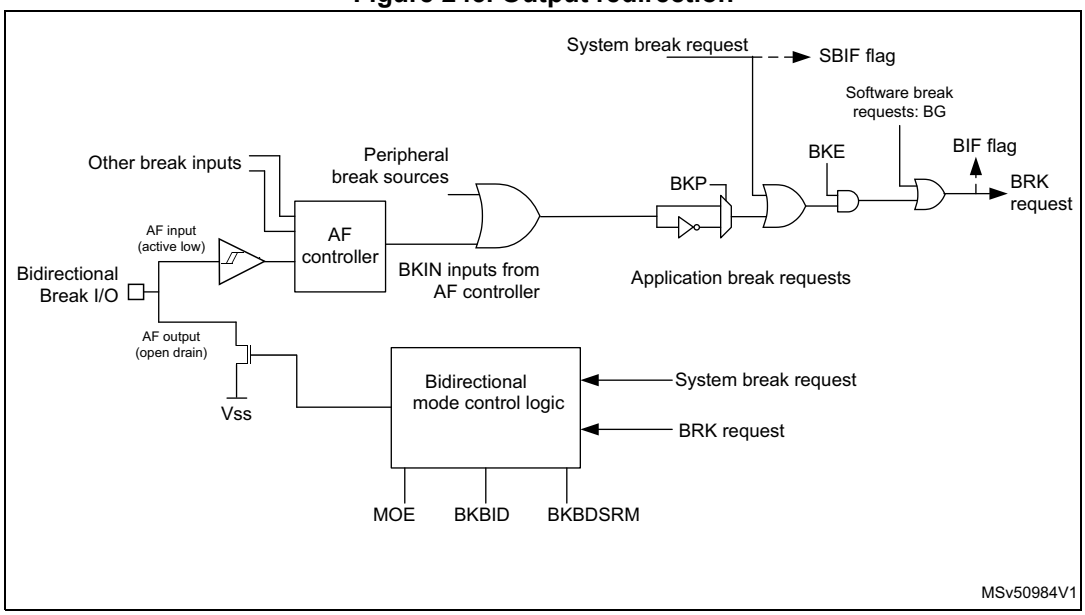
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 245. Output redirection



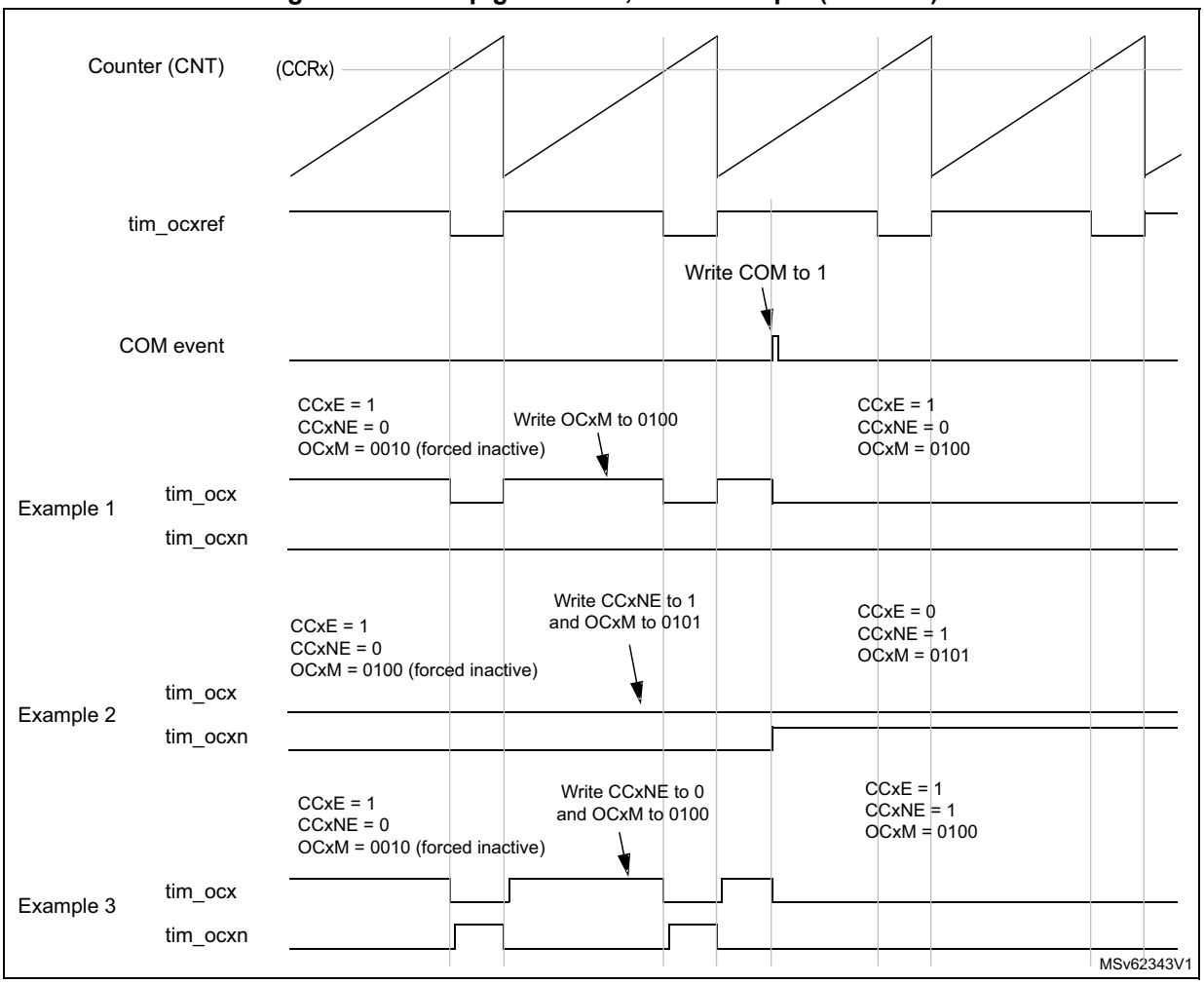
25.3.13 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 246* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 246. 6-step generation, COM example (OSSR=1)



25.3.14 One-pulse mode

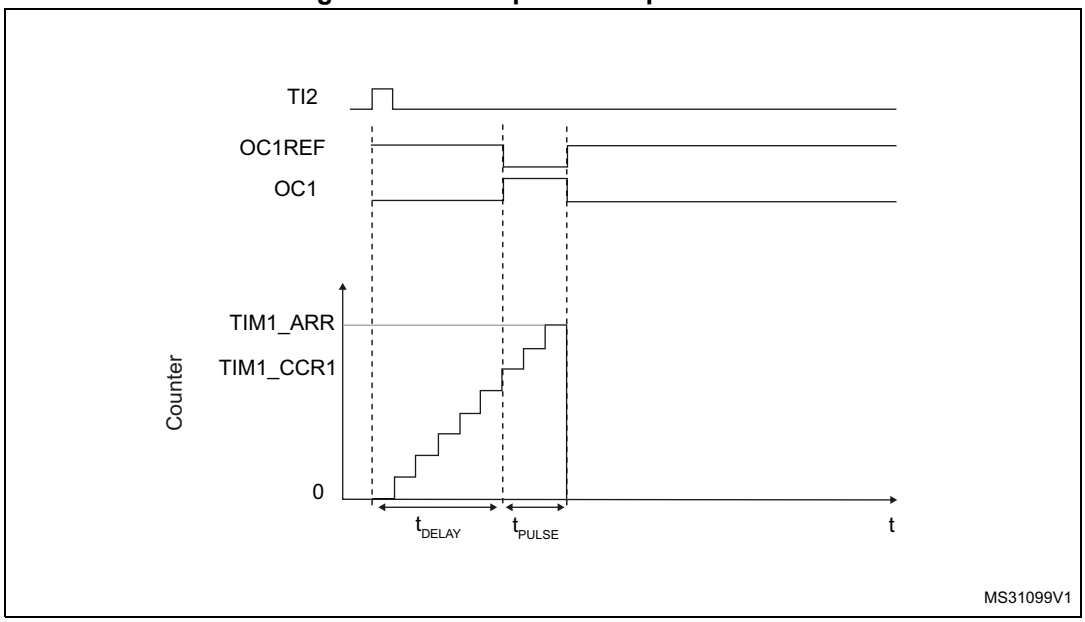
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 247. Example of one pulse mode



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

25.3.15 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag, to be atomically read. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

25.3.16 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

25.3.17 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

25.3.18 Using timer output as trigger for other timers (TIM16/TIM17)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger.

For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

25.3.19 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module. For more details, refer to *Section 36.11.5: DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)*.

For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

25.4 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

25.4.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (TIx),
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 * t_{CK\_INT}$
10: $t_{DTS} = 4 * t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

25.4.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

25.4.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

25.4.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

25.4.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

25.4.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

25.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
									rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

All other values: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

25.4.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high
1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 175* for details.

Table 175. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and GPIO control and alternate function registers.

25.4.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

25.4.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

25.4.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 25.3.1: Time-base unit on page 792* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

25.4.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

25.4.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

25.4.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw		rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input BRK in input mode
1: Break input BRK in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)
See OC/OCN enable description for more details (*Section 25.4.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 827*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1; Break inputs (BRK and CCS clock failure event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 25.4.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 827*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.
See OC/OCN enable description for more details (*Section 25.4.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 827*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0] x t_{dtg} with $t_{dtg} = t_{DTS}$
DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x t_{dtg} with $t_{dtg} = 2 \times t_{DTS}$
DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t_{dtg} with $t_{dtg} = 8 \times t_{DTS}$
DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t_{dtg} with $t_{dtg} = 16 \times t_{DTS}$
Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

25.4.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

25.4.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

25.4.17 TIM16 option register 1 (TIM16_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TI1_RMP[1:0]**: Timer 16 input 1 connection
This bit is set and cleared by software.
00: TIM16 TI1 is connected to GPIO
01: TIM16 TI1 is connected to LSI
10: TIM16 TI1 is connected to LSE
11: TIM16 TI1 is connected to RTC wake-up interrupt

25.4.18 TIM16 alternate function register 1 (TIM16_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
				rw	rw	rw							rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active low
1: COMP2 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active low
1: BKIN input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable
This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.
0: COMP2 input disabled
1: COMP2 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable
This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.
0: COMP1 input disabled
1: COMP1 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable
This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.
0: BKIN input disabled
1: BKIN input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

25.4.19 TIM16 input selection register (TIM16_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM16_CH1 input
Others: Reserved

25.4.20 TIM17 option register 1 (TIM17_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TI1_RMP[1:0]**: Timer 17 input 1 connection
This bit is set and cleared by software.
00: TIM17 TI1 is connected to GPIO
01: TIM17 TI1 is connected to MSI
10: TIM17 TI1 is connected to HSE/32
11: TIM17 TI1 is connected to MCO

25.4.21 TIM17 alternate function register 1 (TIM17_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
				rw	rw	rw							rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active low
1: COMP2 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active low
1: BKIN input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable
This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.
0: COMP2 input disabled
1: COMP2 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable
This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.
0: COMP1 input disabled
1: COMP1 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable
This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.
0: BKIN input disabled
1: BKIN input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

25.4.22 TIM17 input selection register (TIM17_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM17_CH1 input
Others: Reserved

25.4.23 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 176. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]				OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0										0	0	0		0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 176. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	**TIMx_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIMx_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DTG[7:0]							
	Reset value				0		0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM16_OR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP [1:0]	
	Reset value																															0	0
0x50	**TIM17_OR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP [1:0]	
	Reset value																															0	0
0x60	**TIM16_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																					0	0	0							0	0	1
0x60	**TIM17_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																					0	0	0							0	0	1
0x68	**TIM16_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x68	**TIM17_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.4 on page 62* for the register boundary addresses.

26 Low-power timer (LPTIM)

26.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

26.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter

26.3 LPTIM implementation

Table 177 describes LPTIM implementation on STM32WLEx devices. The full set of features is implemented in LPTIM1. LPTIM2 and LPTIM3 support a smaller set of features, but is otherwise identical to LPTIM1.

Table 177. STM32WLEx LPTIM features

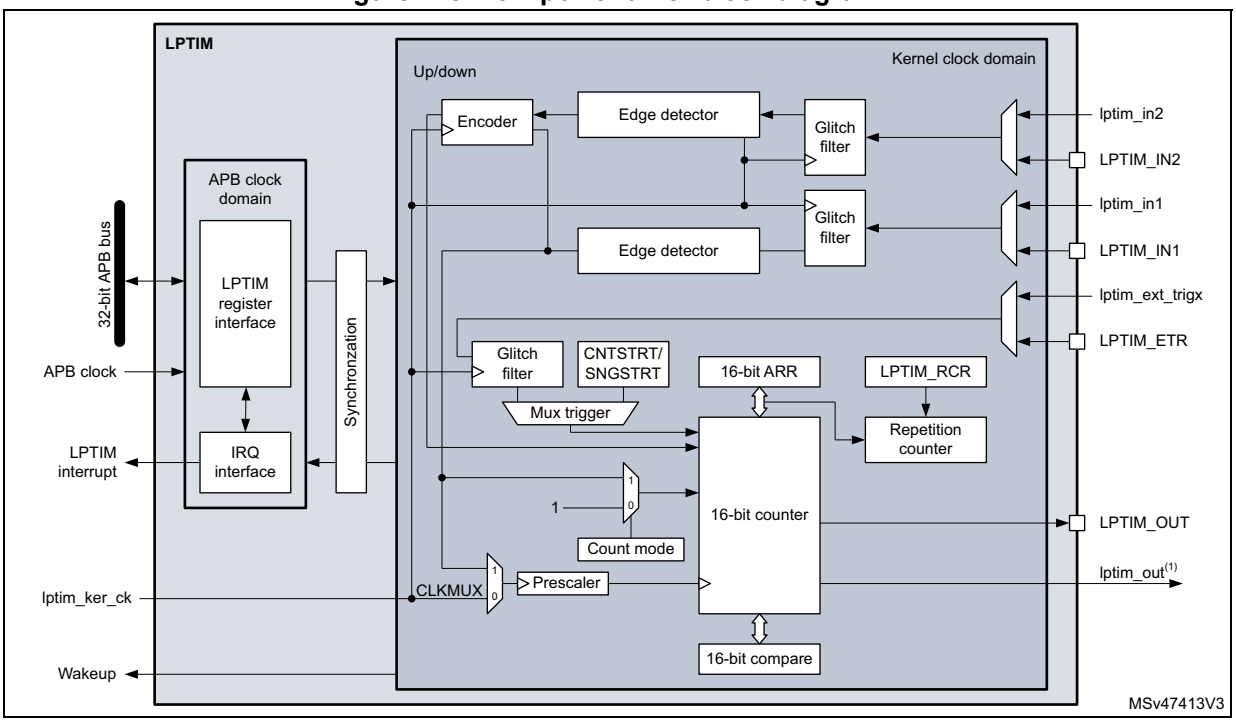
LPTIM modes/features(1)	LPTIM1	LPTIM2	LPTIM3
Encoder mode	X	-	-
External input clock	X	X	X
Wakeup from Stop	(2)	(3)	(3)

1. X = supported.
2. Wakeup supported from Stop 0, Stop 1 and Stop 2 modes.
3. Wakeup supported from Stop 0 and Stop 1 modes.

26.4 LPTIM functional description

26.4.1 LPTIM block diagram

Figure 248. Low-power timer block diagram(a)



1. lptim_out is the internal LPTIM output signal that can be connected to internal peripherals.

a. LPTIM2/LPTIM3 has only the input channel 1, no input channel 2.

26.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 178. LPTIM input/output pins

Names	Signal type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM Output GPIO pin

Table 179. LPTIM internal signals

Names	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1	Digital input	Internal LPTIM input 1
lptim_in2	Digital input	Internal LPTIM input 2(1)
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wakeup event

1. Only applies to LPTIM1

26.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter:

Table 180. LPTIM1 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM1_ETR alternate function
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B
lptim_ext_trig3	TAMP1 input detection
lptim_ext_trig4	TAMP2 input detection
lptim_ext_trig5	TAMP3 input detection
lptim_ext_trig6	comp1_out
lptim_ext_trig7	comp2_out

Table 181. LPTIM2 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM2_ETR alternate function
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B
lptim_ext_trig3	TAMP1 input detection
lptim_ext_trig4	TAMP2 input detection
lptim_ext_trig5	TAMP3 input detection
lptim_ext_trig6	comp1_out
lptim_ext_trig7	comp2_out

Table 182. LPTIM3 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM3_ETR alternate function
lptim_ext_trig1	lptim1_out
lptim_ext_trig2	lptim2_out
lptim_ext_trig3	-
lptim_ext_trig4	-
lptim_ext_trig5	-
lptim_ext_trig6	-
lptim_ext_trig7	-

Table 183. LPTIM1 input 1 connection

lptim_in1	LPTIM1 input 1 connected to
lptim_in1	GPIO pin as LPTIM1_IN1 alternate function
lptim_in1	comp1_out

Table 184. LPTIM1 input 2 connection

lptim_in2	LPTIM1 input 2 connected to
lptim_in2	GPIO pin as LPTIM1_IN2 alternate function
lptim_in2	comp2_out

Table 185. LPTIM2 input 1 connection

lptim_in1	LPTIM2 input 1 connected to
lptim_in1	GPIO pin as LPTIM2_IN1 alternate function
lptim_in1	comp1_out
lptim_in1	comp2_out
lptim_in1	comp1_out or comp2_out

Table 186. LPTIM3 input 1 connection

lptim_in1	LPTIM3 input 1 connected to
lptim_in1	GPIO pin as LPTIM3_IN1 alternate function
lptim_in1	comp1_out
lptim_in1	comp2_out
lptim_in1	comp1_out or comp2_out

26.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal should also be provided (first configuration). In this case, the internal clock signal frequency should be at least four times higher than the external clock signal frequency.

26.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source should first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

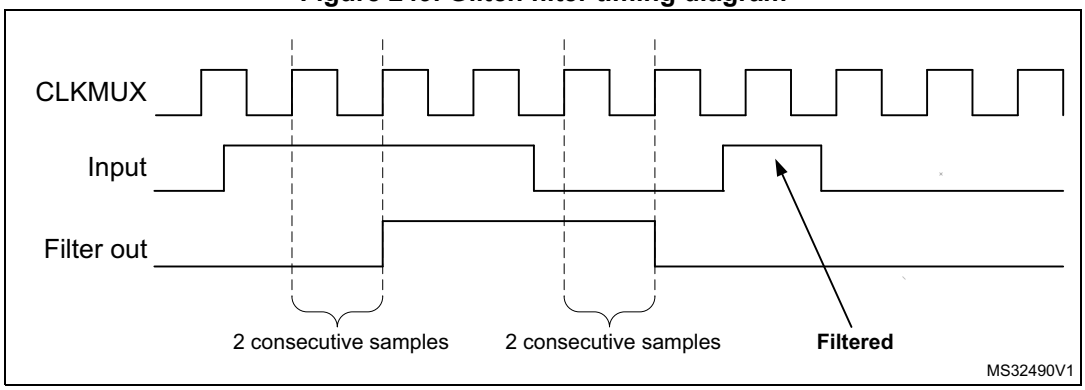
The digital filters are divided into two groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that should be detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 249* shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

Figure 249. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT and TRGFLT bits to '0'. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.

26.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

Table 187. Prescaler division ratios

programming	dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

26.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

26.4.8 Operating mode

The LPTIM features two operating modes:

- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

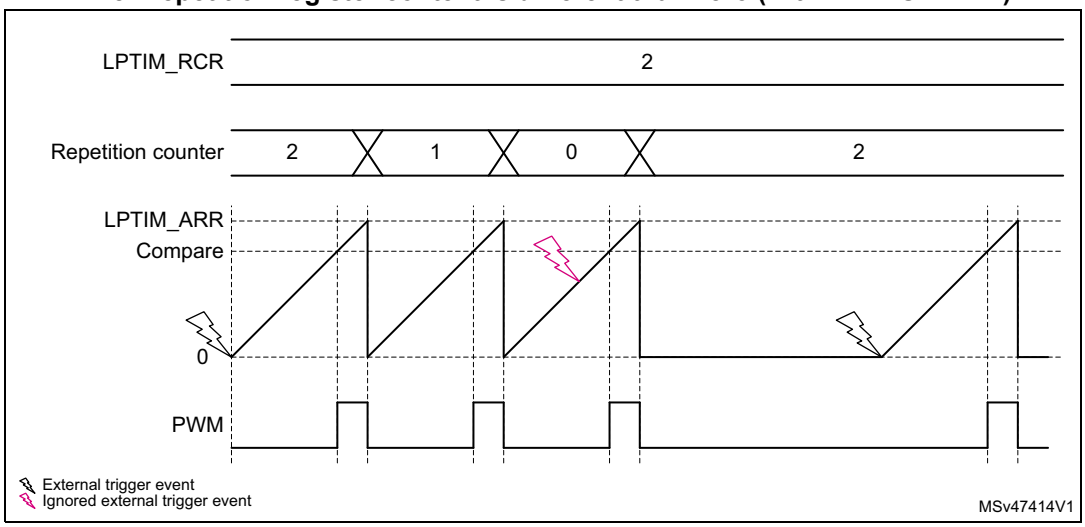
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 250*.

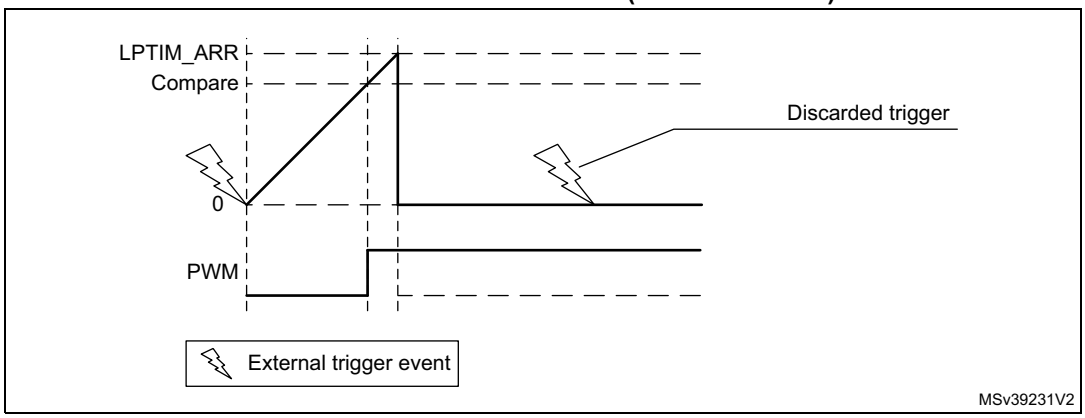
Figure 250. LPTIM output waveform, single counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

It should be noted that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 251*.

Figure 251. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = ‘00’), the SNGSTRT setting starts the counter for one-shot counting.

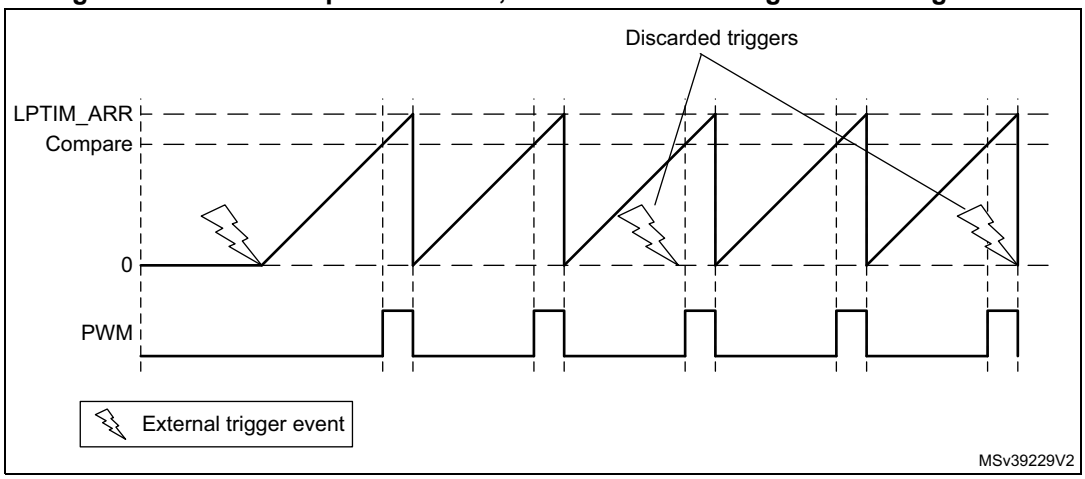
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 252*.

In case of software start (TRIGEN[1:0] = ‘00’), setting CNTSTRT starts the counter for continuous counting.

Figure 252. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to ‘1’). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

26.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

26.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CMP (compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CMP. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT registers.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM_ARR register value be strictly greater than the LPTIM_CMP register value.

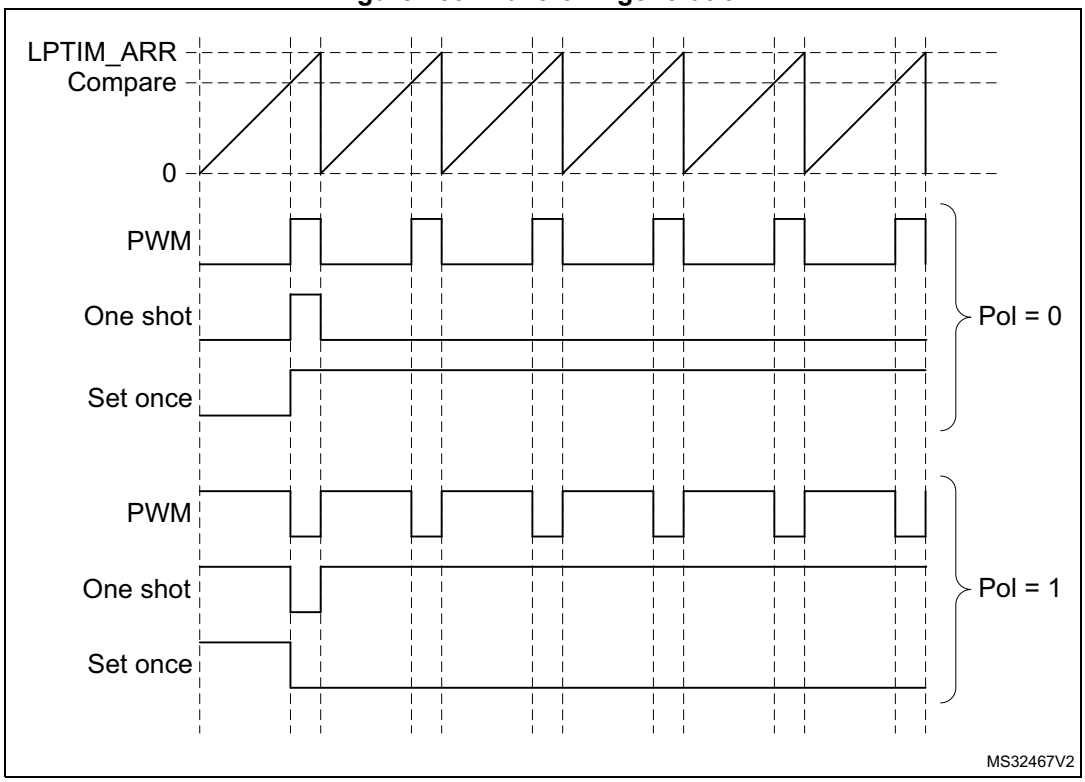
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to ‘0’ forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to ‘1’ forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. *Figure 253* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL bit.

Figure 253. Waveform generation



26.4.11 Register update

The LPTIM_ARR register and LPTIM_CMP register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR and the LPTIM_CMP registers are updated:

- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR and the LPTIM_CMP registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR and the LPTIM_CMP registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag and the CMPOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register and the LPTIM_CMP register.

After a write to the LPTIM_ARR register or the LPTIM_CMP register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag or the CMPOK flag be set, leads to unpredictable results.

26.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

26.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM_CFGR and LPTIM_IER registers must be modified only when the LPTIM is disabled.

26.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.
- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset should be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset should be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  It should be noted that to read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So developer should make sure that these two mechanisms are used exclusively.

26.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two Down and Up flags in the LPTIM_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to ‘1’. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder’s position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

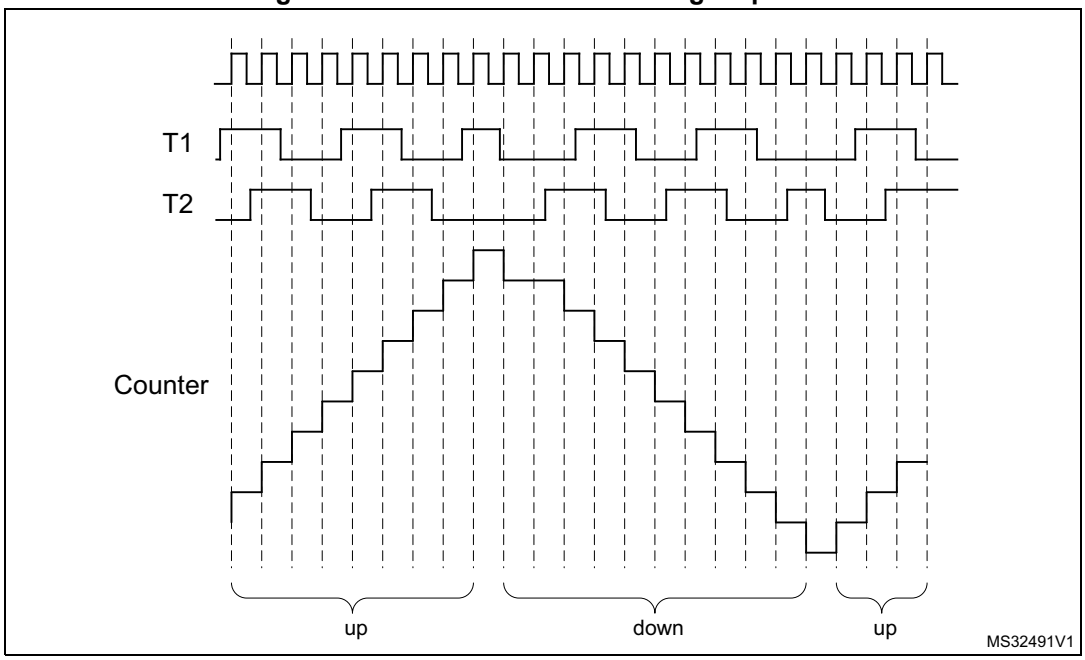
Table 188. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to ‘0’. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be ‘000’).

Figure 254. Encoder mode counting sequence



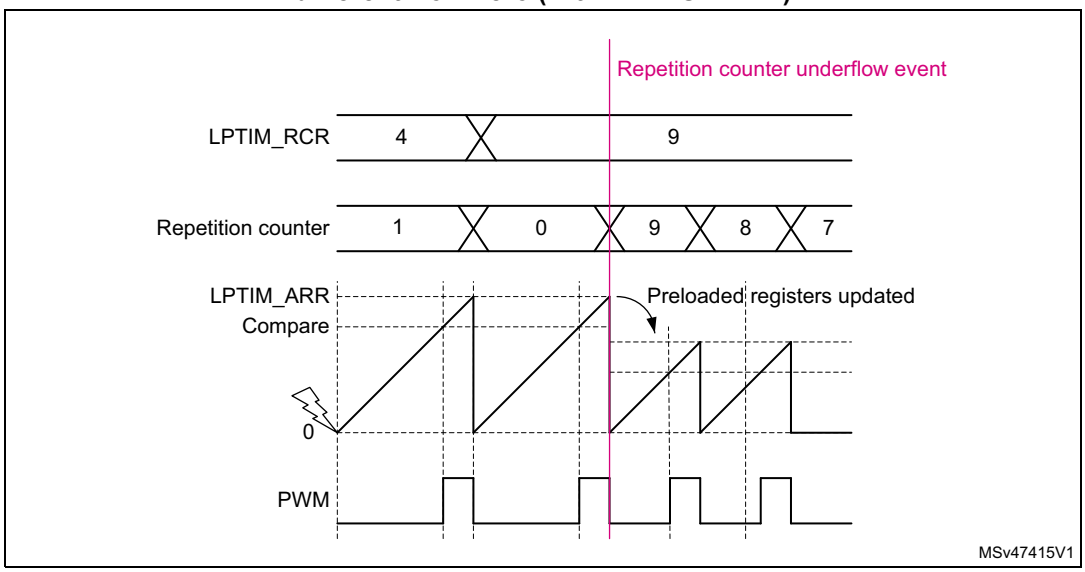
26.4.16 Repetition Counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 255*.

Figure 255. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to section "Register update" for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_IER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, that new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain so that the new written value is loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_IER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the REPOK flag is set, leads to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

26.4.17 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the DBG_LPTIM_STOP configuration bit in the DBG module.

26.5 LPTIM low-power modes

Table 189. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode (refer to *Section 26.3: LPTIM implementation*).
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

26.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_IER register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK

Note: If any bit in the LPTIM_IER register is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.

Table 190. Interrupt events

Interrupt event	Description
Compare match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the compare register (LPTIM_CMP).
Auto-reload match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the Auto-reload register (LPTIM_ARR).
External trigger event	Interrupt flag is raised when an external trigger event is detected
Auto-reload register update OK	Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete.
Compare register update OK	Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete.
Direction change	Used in Encoder mode. Two interrupt flags are embedded to signal direction change: – UP flag signals up-counting direction change – DOWN flag signals down-counting direction change.

Table 190. Interrupt events (continued)

Interrupt event	Description
Update Event	Interrupt flag is raised when the repetition counter underflows (or contains zero) and the LPTIM counter overflows.
Repetition register update Ok	REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed.

26.7 LPTIM registers

26.7.1 LPTIM interrupt and status register (LPTIM_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP OK	UE	DOWN	UP	ARR OK	CMP OK	EXT TRIG	ARRM	CMPM
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOK**: Repetition register update Ok

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 26.3: LPTIM implementation.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 26.3: LPTIM implementation.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMPOK**: Compare register update OK

CMPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_CMP register has been successfully completed.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CMPM**: Compare match

The CMPM bit is set by hardware to inform application that LPTIM_CNT register value reached the LPTIM_CMP register's value.

26.7.2 LPTIM interrupt clear register (LPTIM_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
							w	w	w	w	w	w	w	w	w

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 26.3.

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 26.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag

Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMPOKCF**: Compare register update OK clear flag

Writing 1 to this bit clears the CMPOK flag in the LPTIM_ISR register

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CMPMCF**: Compare match clear flag
Writing 1 to this bit clears the CMP flag in the LPTIM_ISR register

26.7.3 LPTIM interrupt enable register (LPTIM_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 26.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 26.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMPOKIE**: Compare register update OK Interrupt Enable
0: CMPOK interrupt disabled
1: CMPOK interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
- 0: EXTTRIG interrupt disabled
- 1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
- 0: ARRM interrupt disabled
- 1: ARRM interrupt enabled

Bit 0 **CMPMIE**: Compare match Interrupt Enable
- 0: CMPM interrupt disabled
- 1: CMPM interrupt enabled

Caution: The LPTIM_IER register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0')

26.7.4 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	WAV POL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable

The ENC bit controls the Encoder mode
- 0: Encoder mode disabled
- 1: Encoder mode enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 26.3.

Bit 23 **COUNTMODE**: counter mode enabled

The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
- 0: the counter is incremented following each internal clock pulse
- 1: the counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode

The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CMP registers update modality
- 0: Registers are updated after each APB bus write access
- 1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity

The WAVPOL bit controls the output polarity

0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers

1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers

Bit 20 **WAVE**: Waveform shape

The WAVE bit controls the output shape

0: Deactivate Set-once mode

1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable

The TIMOUT bit controls the Timeout feature

0: A trigger event arriving when the timer is already started is ignored

1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity

The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:

00: software trigger (counting start is initiated by software)

01: rising edge is the active edge

10: falling edge is the active edge

11: both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0

001: lptim_ext_trig1

010: lptim_ext_trig2

011: lptim_ext_trig3

100: lptim_ext_trig4

101: lptim_ext_trig5

110: lptim_ext_trig6

111: lptim_ext_trig7

See *Section 26.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1

001: /2

010: /4

011: /8

100: /16

101: /32

110: /64

111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any trigger active level change is considered as a valid trigger

01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.

10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.

11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any external clock signal level change is considered as a valid transition

01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock Polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: the rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 1 is active.

01: the falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 2 is active.

10: both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 3 is active.

11: not allowed

Refer to *Section 26.4.15: Encoder mode* for more details about Encoder mode sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

26.7.5 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.

This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software should consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled.

1: LPTIM is enabled

26.7.6 LPTIM compare register (LPTIM_CMP)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP[15:0]:** Compare value
CMP is the compare value used by the LPTIM.

Caution: The LPTIM_CMP register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

26.7.7 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CMP[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

26.7.8 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value

When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.

26.7.9 LPTIM1 option register (LPTIM1_OR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OR_1**: Option register bit 1

0: LPTIM1 input 2 is connected to I/O

1: LPTIM1 input 2 is connected to COMP2_OUT

Bit 0 **OR_0**: Option register bit 0

0: LPTIM1 input 1 is connected to I/O

1: LPTIM1 input 1 is connected to COMP1_OUT

26.7.10 LPTIM2 option register (LPTIM2_OR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **OR_[1:0]:**

00: input 1 is connected to I/O
01: input 1 is connected to COMP1_OUT
10: input 1 is connected to COMP2_OUT
11: input 1 is connected to COMP1_OUT OR COMP2_OUT

26.7.11 LPTIM3 option register (LPTIM3_OR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **OR_[1:0]:**

00: input 1 is connected to I/O
01: input 1 is connected to COMP1_OUT
10: input 1 is connected to COMP2_OUT
11: input 1 is connected to COMP1_OUT OR COMP2_OUT

26.7.12 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition register value

REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

26.7.13 LPTIM register map

The following table summarizes the LPTIM registers.

Table 191. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN(1)	UP(1)	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
	Reset value																								0	0	0	0	0	0	0	0	0
0x004	LPTIM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF(1)	UPCF(1)	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
	Reset value																								0	0	0	0	0	0	0	0	0
0x008	LPTIM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE(1)	UPIE(1)	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
	Reset value																								0	0	0	0	0	0	0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(1)	COUNTMODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]			Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0		0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	LPTIM1_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
	Reset value																															0	0
0x020	LPTIM2_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
	Reset value																															0	0

Table 191. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	LPTIM3_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
	Reset value																															0	0
0x028	LPTIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

1. *If LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 26.3: LPTIM implementation.*

Refer to *Section 2.4 on page 62* for the register boundary addresses.

27 Infrared interface (IRTIM)

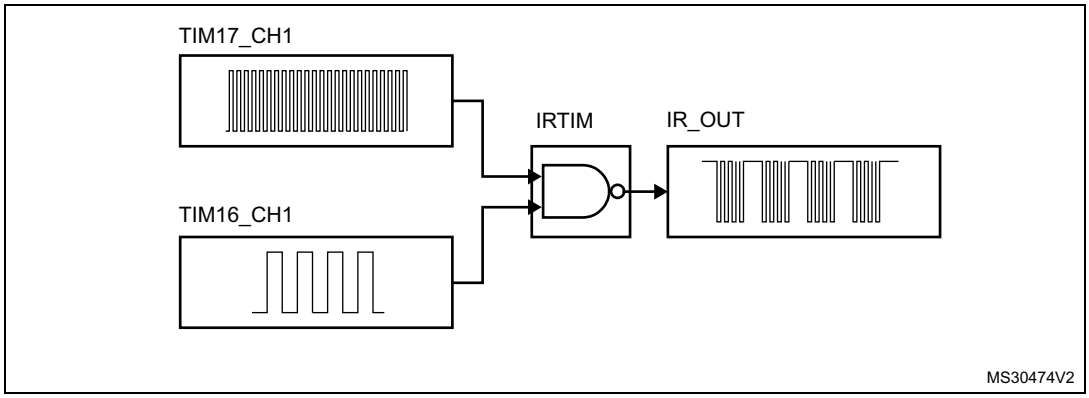
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections withTIM16 and TIM17 as shown in *Figure 256*.

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16_OC1) and TIM17 channel 1 (TIM17_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

Figure 256. IRTIM internal hardware connections with TIM16 and TIM17



All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 generates the modulation envelope.

The infrared function is output on the IR_OUT pin. The activation of this function is done through the GPIOx_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 pin) can be activated through the I2C_PB9_FMP bit in the SYSCFG_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

28 Independent watchdog (IWDG)

28.1 Introduction

The devices feature an embedded watchdog peripheral that offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral detects and solves malfunctions due to software failure, and triggers system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to *Section 29 on page 881*.

28.2 IWDG main features

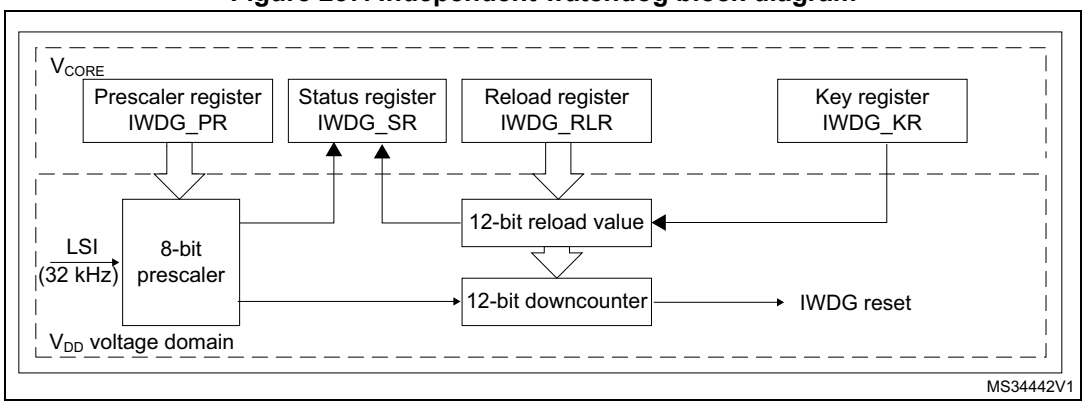
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

28.3 IWDG functional description

28.3.1 IWDG block diagram

Figure 257 shows the functional blocks of the independent watchdog module.

Figure 257. Independent watchdog block diagram



1. The register interface is located in the V_{CORE} voltage domain. The watchdog function is located in the V_{DD} voltage domain, still functional in Stop and Standby mode.

When the independent watchdog is started by writing the value 0x0000 CCCC in the *IWDG key register (IWDG_KR)*, the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

Once running, the IWDG cannot be stopped.

28.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than the value stored in the *IWDG window register (IWDG_WINR)*, then a reset is provided.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the *IWDG reload register (IWDG_RLR)* value and ease the cycle number calculation to generate the next reload.

Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically refreshes the counter value in the *IWDG reload register (IWDG_RLR)*.

Note: Writing the window value allows the counter value to be refreshed by the RLR when IWDG status register (IWDG_SR) is set to 0x0000 0000.

Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA).

28.3.3 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the *IWDG key register (IWDG_KR)* is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

28.3.4 Low-power freeze

Depending on the IWDG_STOP and IWDG_STBY options configuration, the IWDG can continue counting or not during the Stop mode and the Standby mode, respectively. If the IWDG is kept running during Stop or Standby modes, it can wake up the device from this mode. Refer to *User and read protection option bytes* for more details.

28.3.5 Register access protection

Write access to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)* and *IWDG window register (IWDG_WINR)* is protected. To modify them, the user must first write the code 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or of the downcounter reload value or of the window value is ongoing.

28.3.6 Debug mode

When the device enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on the configuration of the corresponding bit in DBGMCU freeze register.

28.4 IWDG registers

Refer to *Section 1.2 on page 55* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

28.4.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 0x5555 to enable access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers (see *Section 28.3.5: Register access protection*)

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

28.4.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see *Section 28.3.5: Register access protection*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset in order to be able to change the prescaler divider.

000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

28.4.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Register access protection*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet for the timeout information.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

28.4.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WVU	RVU	PVU
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five cycles).

Window value can be updated only when WVU bit is reset.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to five cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.

28.4.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 28.3.5*, they contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

28.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

Table 192. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**IWDG_KR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PR[2:0]		
	Reset value																														0	0	0
0x08	**IWDG_RLR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WVU	RVU	PVU
	Reset value																														0	0	0
0x10	**IWDG_WINR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.4 on page 62* for the register boundary addresses.

29 System window watchdog (WWDG)

29.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications which require the watchdog to react within an accurate timing window.

29.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 259*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40.

29.3 WWDG functional description

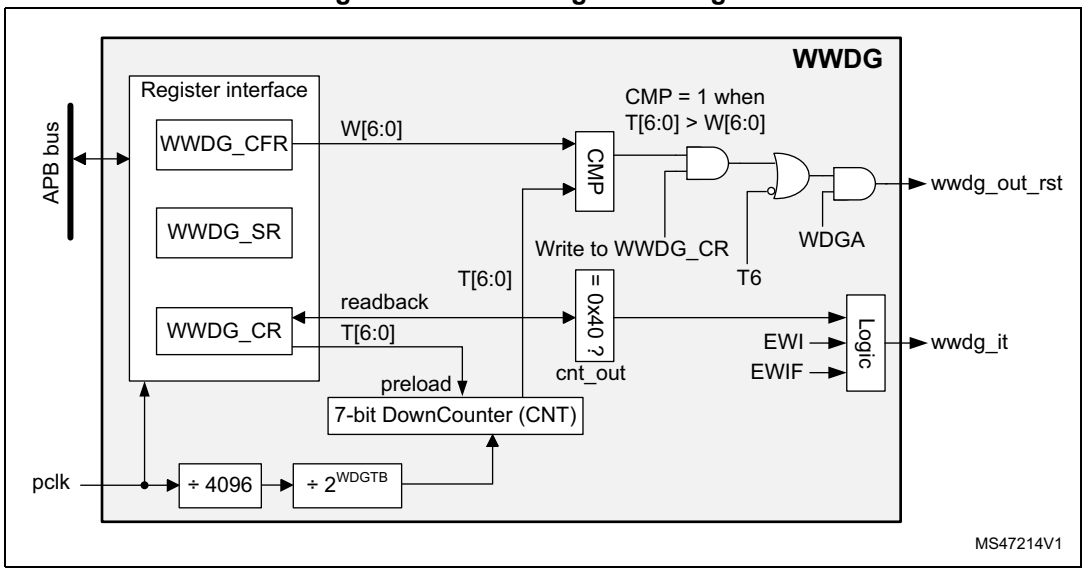
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 258* for the WWDG block diagram.

29.3.1 WWDG block diagram

Figure 258. Watchdog block diagram



29.3.2 WWDG internal signals

Table 193 gives the list of WWDG internal signals.

Table 193. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

29.3.3 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

29.3.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 259*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than the window register value and greater than 0x3F. *Figure 259* describes the window watchdog process.

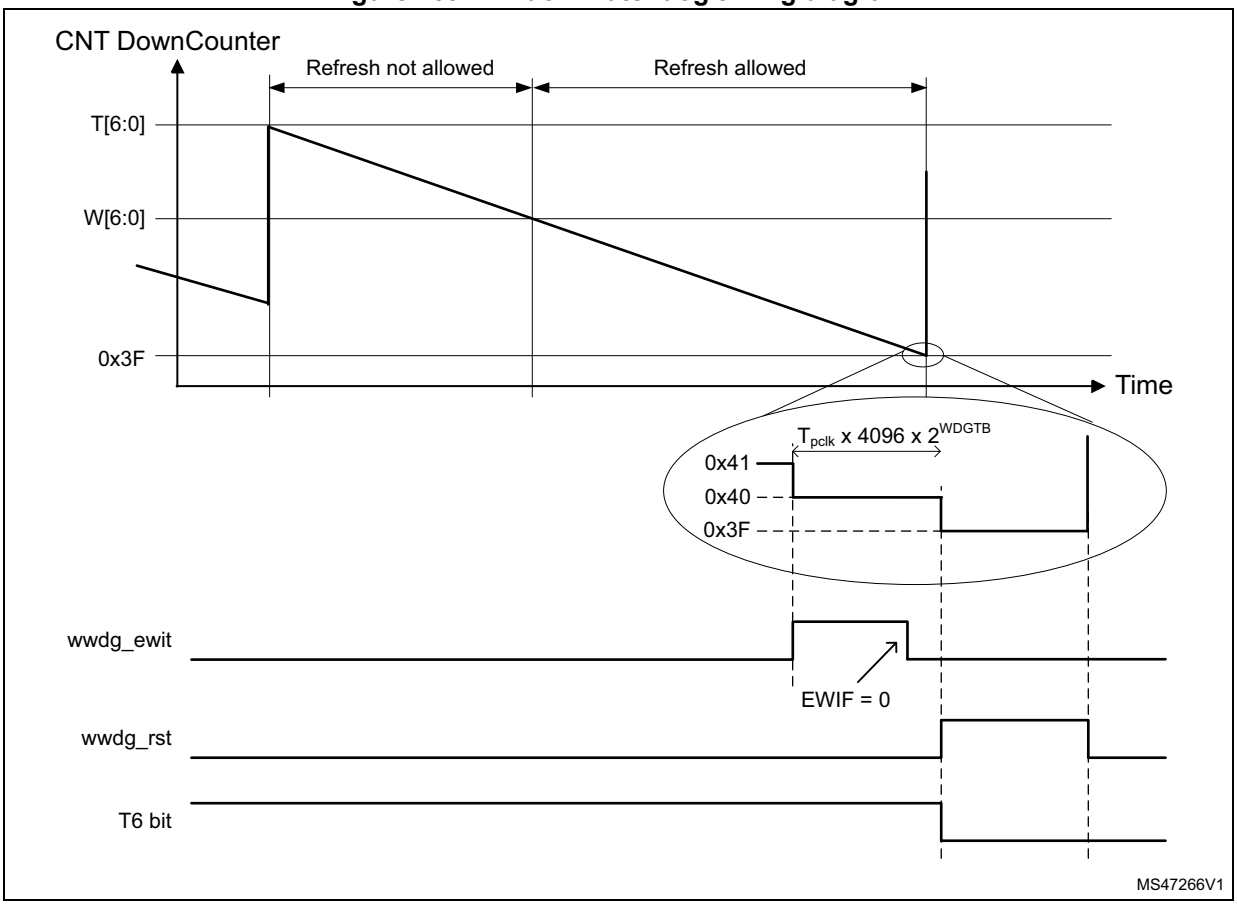
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

29.3.5 How to program the watchdog timeout

Use the formula in *Figure 259* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 259. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

t_{WWDG}: WWDG timeout

t_{PCLK}: APB clock period measured in ms

4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3 and T[5:0] is set to 63:

$$t_{WWDG} = (1/48000) \times 4096 \times 2^3 \times (63+1) = 43.69\text{ms}$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

29.3.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to *Section 36: Debug support (DBG)*.

29.4 WWDG interrupts

The early wakeup interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the down-counter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging) before resetting the device.

In some applications the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR has to reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

Note: When the EWI interrupt cannot be served (for example due to a system lock in a higher priority task) the WWDG reset is eventually generated.

29.5 WWDG registers

Refer to *Section 1.2 on page 55* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

29.5.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

29.5.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

29.5.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

29.5.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 194. WWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**WWDG_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	**WWDG_CFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	**WWDG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.4 on page 62* for the register boundary addresses.

30 Real-time clock (RTC)

30.1 Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in V_{BAT} mode.

30.2 RTC main features

The RTC supports the following features (see *Figure 260: RTC block diagram*):

- Calendar with subsecond, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wakeup timer (WUT) for periodic events with programmable resolution and period.

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the V_{BAT} pin.

The RTC clock sources can be:

- A 32.768 kHz external crystal (LSE)
- An external resonator or oscillator (LSE)
- The internal low power RC oscillator (LSI, with typical frequency of 32 kHz)
- The high-speed external clock (HSE), divided by a prescaler in the RCC.

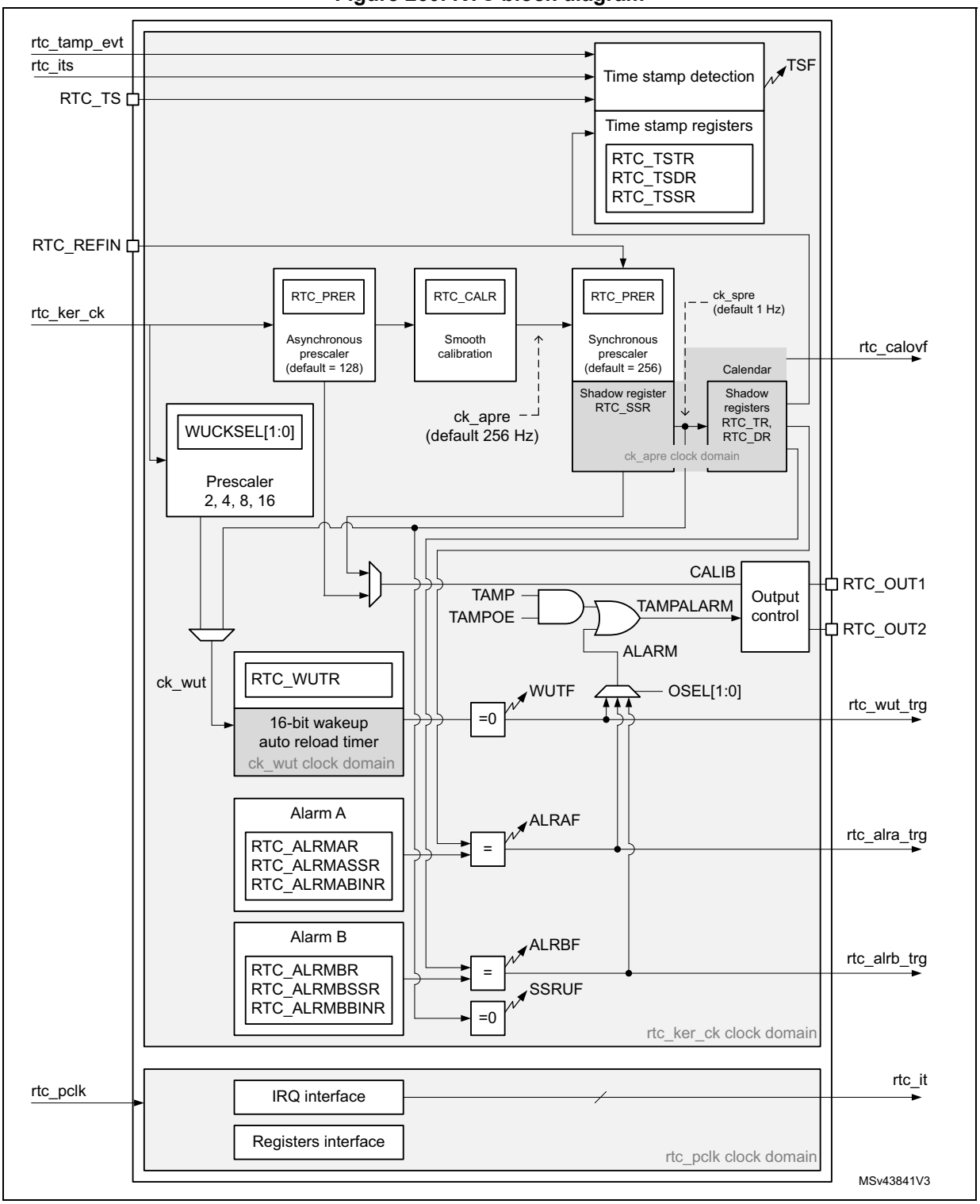
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE. When clocked by the LSI, the RTC is not functional in V_{BAT} mode, but is functional in all low-power modes except Shutdown mode.

All RTC events (Alarm, WakeUp Timer, Timestamp) can generate an interrupt and wakeup the device from the low-power modes.

30.3 RTC functional description

30.3.1 RTC block diagram

Figure 260. RTC block diagram



30.3.2 RTC pins and internal signals

Table 195. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 select one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between TAMP and ALARM outputs.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wakeup outputs. TAMP is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 196. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_it	Output	RTC interrupts (refer to *Section 30.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wakeup timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or V_{BAT} when the selected clock is not LSE. Refer to *Section 30.4: RTC low-power modes* for more details.

Table 197. RTC interconnection

Signal name	Source/destination
rtc_its	From power controller (PWR): main power loss/switch to V_{BAT} detection output
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_calovf	To TAMP peripheral: tamp_itamp5

The triggers outputs can be used as triggers for other peripherals.

30.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

Both RTC and TAMP peripherals provide functions on these I/Os (refer to *Section 31: Tamper and backup registers (TAMP)*).

RTC_OUT1, RTC_TS and TAMP_IN1 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V_{BAT} mode.

The output mechanism follows the priority order shown in *Table 198*.

Table 198. PC13 configuration(1)

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Push-Pull	01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care
	00	1						
	01 or 10 or 11	1						

Table 198. PC13 configuration(1) (continued)

PC13 Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Open-Drain(2)	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care
TAMP_IN1 input floating		00	0	0	Don't care	Don't care	Don't care	1	0
		00	0	1	1				
		Don't care	Don't care	0					
RTC_TS and TAMP_IN1 input floating		00	0	0	Don't care	Don't care	Don't care	1	1
		00	0	1	1				
		Don't care	Don't care	0					
RTC_TS input floating		00	0	0	Don't care	Don't care	Don't care	0	1
		00	0	1	1				
		Don't care	Don't care	0					

Table 198. PC13 configuration(1) (continued)

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
Wakeup pin or Standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0
	00	0	1	1				
	Don't care	Don't care	0					

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

In addition, it is possible to output RTC_OUT2 on PA4 pin thanks to OUT2EN bit. This output is not available in V_{BAT} mode. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in table *Table 199*.

Table 199. RTC_OUT mapping

OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1 on PC13	RTC_OUT2 on PA4
00	0	0	-	-
00	1		CALIB	-
01 or 10 or 11	Don't care		TAMPALRM	-
00	0	1	-	-
00	1		-	CALIB
01 or 10 or 11	0		-	TAMPALRM
01 or 10 or 11	1		TAMPALRM	CALIB

30.3.4 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to "Reset and clock control (RCC)".

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 260: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 30.3.8: Periodic auto-wakeup* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are

used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

30.3.5 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 30.6.10: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

30.3.6 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : Calibration ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

30.3.7 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRMABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

30.3.8 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this allows the wakeup interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, this allows a wakeup time to be achieved from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wakeup timer on page 898*), the timer starts counting down. When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000<WUTOCLR<=WUT).

The wakeup flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 30.3.1: RTC block diagram*).

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

30.3.9 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the sub-second field SS[15:0], clocked by ck_apre, allowing the generation of a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD = 0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bits INIT, BIN and BCDU in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.

If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.

3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRMABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode.
   If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wakeup auto-reload value WUT[15:0], WUTOCLR[15:0] and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting.
   If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

30.3.10 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 896*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 30.3.12: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

30.3.11 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wakeup timer register (RTC_WUTR), and the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

30.3.12 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

30.3.13 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

30.3.14 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

Calibration ultra-low-power mode

The calibration consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL=0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL=0, in order to have a 32 second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

30.3.15 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a tamper event is detected on the TAMP_INx pinx.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the V_{BAT} supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set 2 ck_apre cycles after the timestamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

Optionally, a tamper event can cause a timestamp to be recorded. See the description of the TAMPTS control bit in the RTC control register (RTC_CR).

30.3.16 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV\_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When the CALIB output is selected, the RTC_OUT1 pin is automatically configured but the RTC_OUT2 pin must be set as alternate function.

When COSEL is cleared, the CALIB output is the output of the 6^{th} stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8^{th} stage of the synchronous prescaler.

30.3.17 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set is the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wakeup flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

When the TAMPALRM output is selected, the RTC_OUT1 pin is automatically configured but the RTC_OUT2 pin must be set as alternate function. In case the TAMPALRM is configured open-drain in the RTC, the RTC_OUT1 GPIO must be configured as input.

30.4 RTC low-power modes

Table 200. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.
Shutdown	The RTC remains active when the RTC clock source is LSE. RTC interrupts cause the device to exit the Shutdown mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 201. RTC pins functionality over modes

Functions	Functional in all low-power modes except Standby and Shutdown modes	Functional in Standby and Shutdown mode	Functional in V_{BAT} mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	No	No

30.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

Table 202. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode	Exit from Shutdown mode
RTC	Alarm A	ALRAF	ALRAIE	write 1 in CALRAF	Yes	Yes(3)	Yes(4)
	Alarm B	ALRBF	ALRBIE	write 1 in CALRBF	Yes	Yes(3)	Yes(4)
	Timestamp	TSF	TSIE	write 1 in CTSF	Yes	Yes(3)	Yes(4)
	Wakeup timer interrupt	WUTF	WUTIE	write 1 in CWUTF	Yes	Yes(3)	Yes(4)
	SSR underflow	SSRUF	SSRUIE	write 1 in CSSRUF	Yes	Yes(3)	Yes(4)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. Wakeup from Stop and Standby modes is possible only when the RTC clock source is LSE or LSI.
4. Wakeup from Shutdown modes is possible only when the RTC clock source is LSE.

30.6 RTC registers

Refer to *Section 1.2 on page 55* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

30.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 896* and *Reading the calendar on page 898*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 896*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

30.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 896* and *Reading the calendar on page 898*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 896*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

30.6.3 RTC sub second register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values
When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):
SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.
When BCD mode is selected (BIN=00):
SS[31:16] are forced by hardware to 0x0000.

SS[15:0]: Sub second value/Synchronous binary counter LSB values
When Binary mode is selected (BIN = 01 or 10 or 11):
SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.
When BCD mode is selected (BIN=00):
SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:
Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)
SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

30.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 896*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

0x0: 1s calendar increment is generated each time SS[7:0] = 0

0x1: 1s calendar increment is generated each time SS[8:0] = 0

0x2: 1s calendar increment is generated each time SS[9:0] = 0

0x3: 1s calendar increment is generated each time SS[10:0] = 0

0x4: 1s calendar increment is generated each time SS[11:0] = 0

0x5: 1s calendar increment is generated each time SS[12:0] = 0

0x6: 1s calendar increment is generated each time SS[13:0] = 0

0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).

01: Free running Binary mode (BCD mode disabled)

10: Free running BCD calendar and Binary modes

11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Wakeup timer configuration update not allowed except in initialization mode

1: Wakeup timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

30.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 896*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 896*.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

30.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUTOCLR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WUTOCLR[15:0]**: Wakeup auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

30.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 896.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	Res.	Res.	TAMPOE	TAMPTS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

Setting this bit allows the RTC outputs to be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bits 28:27 Reserved, must be kept at reset value.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM

0: The tamper flag is not routed on TAMPALRM
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a RTC timestamp to be saved
1: Save RTC timestamp on tamper detection event
TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set after the tamper flags, therefore if TAMPTS and TSIE are set, it is recommended to disable the tamper interrupts in order to avoid servicing 2 interrupts.

Bit 24 **ITSE**: timestamp on internal event enable

0: internal event timestamp disabled
1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable

This bit enables the CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to TAMPALRM output.
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wakeup output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of TAMPALRM output.
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection

When COE = 1, this bit selects which signal is output on CALIB.

0: Calibration output is 512 Hz

1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 30.3.16: Calibration clock output*.

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable

0: Timestamp interrupt disable

1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable

0: Wakeup timer interrupt disabled

1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable

0: Alarm B interrupt disable

1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled

1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable

0: timestamp disable

1: timestamp enable

Bit 10 **WUTE**: Wakeup timer enable

0: Wakeup timer disabled

1: Wakeup timer enabled

Bit 9 **ALRBE**: Alarm B enable

0: Alarm B disabled

1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable

0: Alarm A disabled

1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable
0: SSR underflow interrupt disabled
1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wakeup clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.
11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

30.6.8 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

30.6.9 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 896*.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 × CALP) - CALM.

Refer to *Section 30.3.14: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to 1, the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 30.3.14: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 30.3.14: RTC smooth digital calibration.

Bit 12 **LPCAL**: Calibration low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode should be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 30.3.14: RTC smooth digital calibration on page 901*.

30.6.10 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 896*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

30.6.11 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

30.6.12 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

30.6.13 RTC timestamp sub second register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 SS[31:0]: Sub second value/Synchronous binary counter values
SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

30.6.14 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don’t care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don’t care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don’t care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don't care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

30.6.15 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRMABINR → SS[31:0] value and is automatically reloaded with 0xFFFF FFFF when reaching RTC_ALRMABINR → SS[31:0].
Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit
0: No comparison on sub seconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.
2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.
...
31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.
From 32 to 63: All 32 SS bits are compared and must match to activate alarm.
Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value
This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.
This field is the mirror of SS[14:0] in the RTC_ALRMABINR, and so can also be read or written through RTC_ALRMABINR.

30.6.16 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don’t care in alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don’t care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don’t care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don’t care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

30.6.17 RTC alarm B sub second register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:4]		MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRMBBINR → SS[31:0] value and is automatically reloaded with 0xFFFF FFFF when reaching RTC_ALRMBBINR → SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRMBBINR, and so can also be read or written through RTC_ALRMBBINR.

30.6.18 RTC status register (RTC_SR)

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag

This flag is set by hardware when the SSR rolls under 0. SSRUF is not set when SSCLR=1.

Bit 5 **ITSF**: Internal timestamp flag

This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.
If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wakeup auto-reload counter reaches WUTOCLR value.
If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared 2 APB clock cycles after setting their corresponding clear bit in the RTC_SCR register.

30.6.19 RTC masked interrupt status register (RTC_MISR)

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow masked flag

This flag is set by hardware when the SSR underflow interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp masked flag

This flag is set by hardware when a timestamp on the internal event occurs and timestampinterrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow masked flag

This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp masked flag

This flag is set by hardware when a timestamp interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wakeup timer masked flag

This flag is set by hardware when the wakeup timer interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B masked flag

This flag is set by hardware when the alarm B interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag

This flag is set by hardware when the alarm A interrupt occurs.

30.6.20 RTC status clear register (RTC_SCR)

Address offset: 0x5C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing ‘1’ in this bit clears the SSRUF in the RTC_SR register.

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wakeup timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

30.6.21 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSRR, and so can also be read or written through RTC_ALRMASSR.

30.6.22 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSRR, and so can also be read or written through RTC_ALRMBSSR.

30.6.23 RTC register map

Table 203. RTC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_TR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**RTC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	**RTC_SSR**	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**RTC_ICSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	**RTC_PRER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	**RTC_CR**	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	Res.	Res.	TAMPOE	TAMPTS	ITSE	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**RTC_WPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_ CALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	**RTC_SHIFTR**	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**RTC_TSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	**RTC_TSDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	**RTC_TSSSR**	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 203. RTC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	**RTC_ALRMAR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**RTC_ ALRMASSR**	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]															
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**RTC_ALRMBR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**RTC_ ALRMBSSR**	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]															
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**RTC_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0	0	0	0	0	0	0
0x54	**RTC_MISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
0x5C	**RTC_SCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0	0	0	0	0	0	0
0x70	**RTC_ ALRABINR**	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	**RTC_ ALRBBINR**	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.4 on page 62* for the register boundary addresses.

31 Tamper and backup registers (TAMP)

31.1 Introduction

20 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. They can be used to store sensitive data as their content is protected by an tamper detection circuit. 3 tamper pins and 4 internal tampers are available for anti-tamper detection. The external tamper pins can be configured for edge detection, or level detection with or without filtering.

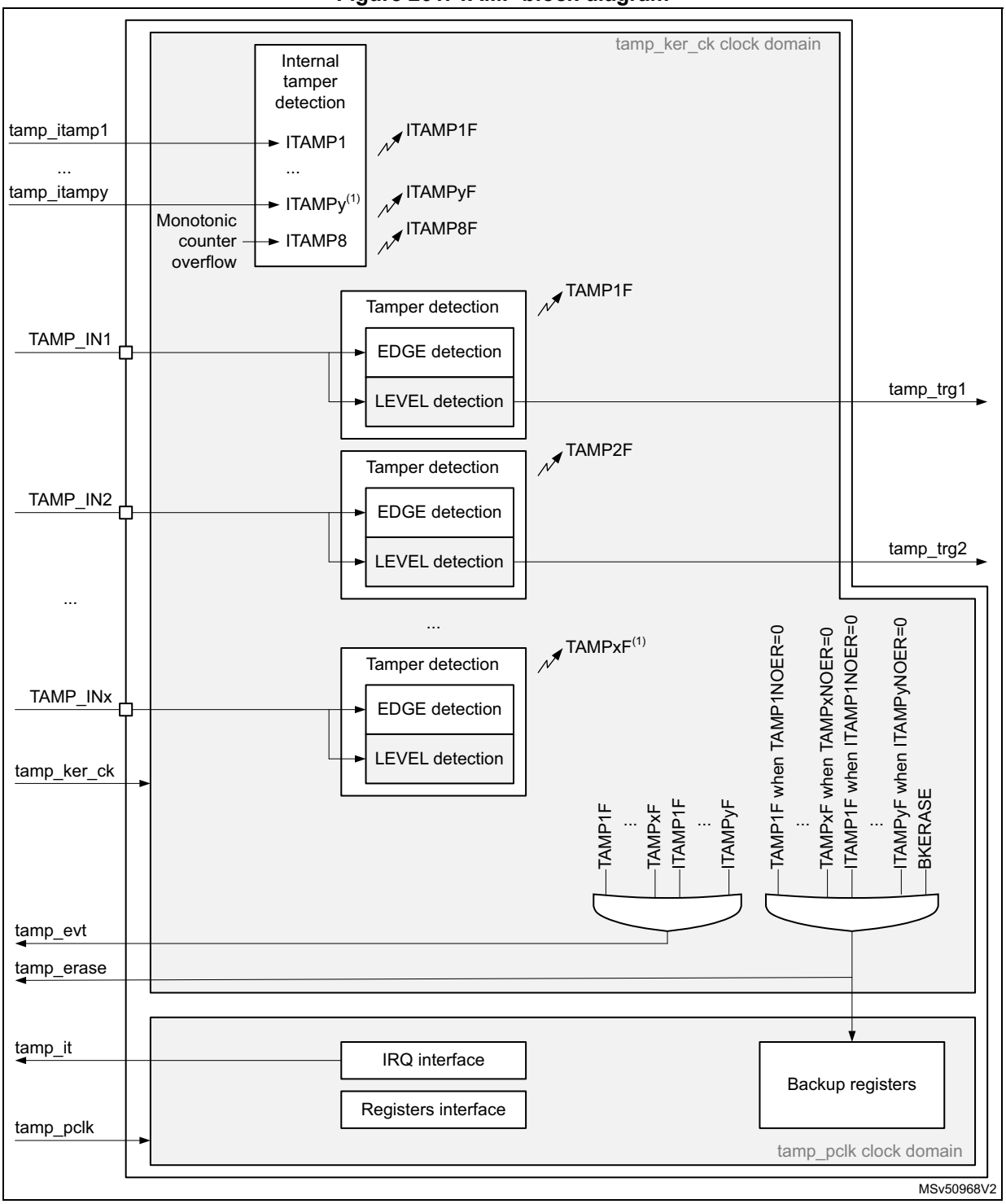
31.2 TAMP main features

- 20 backup registers:
  - the backup registers (TAMP_BKPxR) are implemented in the RTC domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- 3 external tamper detection events.
  - External passive tampers with configurable filter and internal pull-up.
- 4 internal tamper events.
- Any tamper detection can generate a RTC timestamp event.
- Any tamper detection can erase the backup registers, SRAM2 and PKA SRAM.
- Monotonic counter.

31.3 TAMP functional description

31.3.1 TAMP block diagram

Figure 261. TAMP block diagram



1. The number of external and internal tampers depends on products.

31.3.2 TAMP pins and internal signals

Table 204. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin

Table 205. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_evt	Output	Tamper event detection (internal or external) The tamp_evt is used to generate a RTC timestamp event
tamp_erase	Output	Device secrets erase request following either tamper event detection (internal or external) or the software erase request done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 31.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or V_{BAT} when the selected clock is not LSE (refer to *Section 31.4: TAMP low-power modes* for more details.

Table 206. TAMP interconnection

Signal name	Source/Destination
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_erase	The tamp_erase signal is used to erase the device secrets listed hereafter: backup registers, SRAM2 and PKA SRAM
tamp_itamp3	LSE monitoring
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	JTAG access in RDP level 1
tamp_itamp8(1)	Monotonic counter overflow

1. This signal is generated in the TAMP peripheral.

31.3.3 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

31.3.4 Tamper detection

The tamper detection can be configured for the following purposes:

- erase the backup registers and the SRAMs listed in *Table 206: TAMP interconnection* (default configuration)
- generate an interrupt, capable to wakeup from Stop and Standby mode
- generate a hardware trigger for the low-power timers

TAMP backup registers

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs except if the TAMPxNOER bit is set, or if the TAMPxMSK is set in the TAMP_CR2 register, or if the ITAMPxNOER bit is set in the TAMP_CR3 register.

The backup registers and the device secrets erased by tamp_erase signal (refer to *Table 206: TAMP interconnection*) can be reset by software by setting the BKERASE bit in the TAMP_CR2 register.

Note: The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.

Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

When TAMPxMSK is cleared:

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck_apre cycles when TAMPFLT differs from 0x0 (level detection with filtering)
- 3 ck_apre cycles when TAMPTS = 1 (timestamp on tamper event)
- No latency when TAMPFLT = 0x0 (edge detection) and TAMPTS = 0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

When TAMPxMSK is set:

A new tamper occurring on the same pin cannot be detected during the latency described above and 2.5 ck_rtc additional cycles.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit in cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration allows the low-power timers in Stop mode to be trigged automatically, without requiring the system wakeup to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in the *Level detection with filtering on tamper inputs (passive mode)* mode.

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set in the TAMP_SR at the same time that TSF or TSOVF is set in the RTC_SR.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge/high level or a falling edge/low level is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When using the edge detection, it is recommended to check by software the tamper pin level just after enabling the tamper detection (by reading the GPIO registers), and before writing sensitive values in the backup registers, to ensure that an active edge did not occur before enabling the tamper event detection.
When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

31.4 TAMP low-power modes

Table 207. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Stop mode.
Standby	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI.Tamper events cause the device to exit the Standby mode.
Shutdown	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE. Tamper events cause the device to exit the Shutdown mode.

31.5 TAMP interrupts

The interrupt channel is set in the interrupt status register. The interrupt output is also activated.

Table 208. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby modes	Exit from Shutdown mode
TAMP	Tamper x(3)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes	Yes(4)	Yes(5)
TAMP	Internal tamper y(3)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPxF	Yes	Yes(4)	Yes(5)

1. The event flags are in the TAMP_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the TAMP_MISR register.
3. The number of tampers and internal tampers events depend on products.
4. In case of level detection with filtering passive tamper mode, wakeup from Stop and Standby modes is possible only when the TAMP clock source is LSE or LSI.
5. In case of level detection with filtering passive tamper mode, wakeup from Shutdown modes is possible only when the TAMP clock source is LSE.

31.6 TAMP registers

Refer to *Section 1.2 on page 55* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

31.6.1 TAMP control register 1 (TAMP_CR1)

Address offset: 0x00

Backup domain reset value: 0xFFFF 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8E	Res.	ITAMP6E	ITAMP5E	Res.	ITAMP3E	Res.	Res.
								rw		rw	rw		rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3E	TAMP2E	TAMP1E
													rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ITAMP8E**: Internal tamper 8 enable
0: Internal tamper 8 disabled.
1: Internal tamper 8 enabled.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable(1)
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR register and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

31.6.2 TAMP control register 2 (TAMP_CR2)

Address offset: 0x04

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	BK ERASE	Res.	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
					rw	rw	rw	w					rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3 NOER	TAMP2 NOER	TAMP1 NOER
													rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input
0: If TAMPFLT ≠ 00 Tamper 3 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 3 input rising edge and high level triggers a tamper detection event.
1: If TAMPFLT ≠ 00 Tamper 3 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 3 input falling edge and low level triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input
0: If TAMPFLT ≠ 00 Tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 2 input rising edge and high level triggers a tamper detection event.
1: If TAMPFLT ≠ 00 Tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 2 input falling edge and low level triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input
0: If TAMPFLT ≠ 00 Tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 1 input rising edge and high level triggers a tamper detection event.
1: If TAMPFLT ≠ 00 Tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 1 input falling edge and low level triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers(1) erase

Writing '1' to this bit reset the backup registers(1). Writing 0 has no effect. This bit is always read as 0.

Bits 22:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask

0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.

1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers(1) are not erased.

The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers(1) are not erased.

The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers(1) are not erased.

The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3NOER**: Tamper 3 no erase

0: Tamper 3 event erases the backup registers.

1: Tamper 3 event does not erase the backup registers(1).

Bit 1 **TAMP2NOER**: Tamper 2 no erase

0: Tamper 2 event erases the backup registers.

1: Tamper 2 event does not erase the backup registers(1).

Bit 0 **TAMP1NOER**: Tamper 1 no erase

0: Tamper 1 event erases the backup registers.

1: Tamper 1 event does not erase the backup registers(1).

1. The device secrets erased by tamp_erase signal (refer to *Table 206: TAMP interconnection*).

31.6.3 TAMP control register 3 (TAMP_CR3)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8 NOER	Res.	ITAMP6 NOER	ITAMP5 NOER	Res.	ITAMP3 NOER	Res.	Res.
								rw		rw	rw		rw		

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ITAMP8NOER**: Internal Tamper 8 no erase
0: Internal Tamper 8 event erases the backup registers.
1: Internal Tamper 8 event does not erase the backup registers(1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ITAMP6NOER**: Internal Tamper 6 no erase
0: Internal Tamper 6 event erases the backup registers.
1: Internal Tamper 6 event does not erase the backup registers(1).

Bit 4 **ITAMP5NOER**: Internal Tamper 5 no erase
0: Internal Tamper 5 event erases the backup registers.
1: Internal Tamper 5 event does not erase the backup registers(1).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ITAMP3NOER**: Internal Tamper 3 no erase
0: Internal Tamper 3 event erases the backup registers.
1: Internal Tamper 3 event does not erase the backup registers(1).

Bit 1 Reserved, must be kept at reset value.

Bit 0 Reserved, must be kept at reset value.

1. and the device secrets erased by tamp_erase signal (refer to *Table 206: TAMP interconnection*).

31.6.4 TAMP filter control register (TAMP_FLTCR)

Address offset: 0x0C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP PUDIS	TAMPPRCH [1:0]		TAMPFLT [1:0]		TAMPFREQ [2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable
This bit determines if each of the TAMPx pins are precharged before each sample.
0: Precharge TAMP_INx pins before sampling (enable internal pull-up)
1: Disable precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP_INx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP_INx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP_INx inputs.

0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).

0x1: Tamper event is activated after 2 consecutive samples at the active level.

0x2: Tamper event is activated after 4 consecutive samples at the active level.

0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the TAMP_INx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

31.6.5 TAMP interrupt enable register (TAMP_IER)

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8IE	Res.	ITAMP6IE	ITAMP5IE	Res.	ITAMP3IE	Res.	Res.
								rw		rw	rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3IE	TAMP2IE	TAMP1IE
													rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable

0: Internal tamper 8 interrupt disabled.

1: Internal tamper 8 interrupt enabled.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled..

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

31.6.6 TAMP status register (TAMP_SR)

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8F	Res.	ITAMP6F	ITAMP5F	Res.	ITAMP3F	Res.	Res.
								r		r	r		r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3F	TAMP2F	TAMP1F
													r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ITAMP8F**: Internal tamper 8 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6F**: Internal tamper 6 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3F**: TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

31.6.7 TAMP masked interrupt status register (TAMP_MISR)

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8MF	Res.	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.
								r		r	r		r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP2MF	TAMP1MF
													r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ITAMP8MF**: Internal tamper 8 interrupt masked flag
This flag is set by hardware when the internal tamper 8 interrupt is raised.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6MF**: Internal tamper 6 interrupt masked flag
This flag is set by hardware when the internal tamper 6 interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 interrupt masked flag
This flag is set by hardware when the internal tamper 5 interrupt is raised.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3MF**: Internal tamper 3 interrupt masked flag
This flag is set by hardware when the internal tamper 3 interrupt is raised.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3MF**: TAMP3 interrupt masked flag
This flag is set by hardware when the tamper 3 interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 interrupt masked flag
This flag is set by hardware when the tamper 2 interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 interrupt masked flag
This flag is set by hardware when the tamper 1 interrupt is raised.

31.6.8 TAMP status clear register (TAMP_SCR)

Address offset: 0x3C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITAMP8F	Res.	CITAMP6F	CITAMP5F	Res.	CITAMP3F	Res.	Res.
								w		w	w		w		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP3F	CTAMP2F	CTAMP1F
													w	w	w

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

31.6.9 TAMP monotonic counter register (TAMP_COUNTR)

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COUNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COUNT[31:0]:**

This register is read-only only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.

31.6.10 TAMP backup x register (TAMP_BKPxR)

Address offset: 0x100 + 0x04 * x, (x = 0 to 19)

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 **BKP[31:0]:**

The application can write or read data to and from these registers.

They are powered-on by V_{BAT} when V_{DD} is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode.

In the default configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

31.6.11 TAMP register map

Table 209. TAMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TAMP_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8E	Res.	ITAMP6E	ITAMP5E	Res.	ITAMP3E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3E	TAMP2E	TAMP1E
	Reset value									1		1	1		1																0	0	0
0x04	**TAMP_CR2**	Res.	Res.	Res.	Res.	Res.	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	Res.	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3NOER	TAMP2NOER	TAMP1NOER
	Reset value						0	0	0	0					0	0	0														0	0	0
0x08	**TAMP_CR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8NOER	Res.	ITAMP6NOER	ITAMP5NOER	Res.	ITAMP3NOER	Res.	Res.
	Reset value																									0		0	0		0		
0x0C	**TAMP_FLTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x2C	**TAMP_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8IE	Res.	ITAMP6IE	ITAMP5IE	Res.	ITAMP3IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value									0		0	0		0																0	0	0
0x30	**TAMP_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8F	Res.	ITAMP6F	ITAMP5F	Res.	ITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3F	TAMP2F	TAMP1F
	Reset value									0		0	0		0																0	0	0
0x34	**TAMP_MISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8MF	Res.	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value									0		0	0		0																0	0	0
0x3C	**TAMP_SCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITAMP8F	Res.	CITAMP6F	CITAMP5F	Res.	CITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value									0		0	0		0																0	0	0
0x40	**TAMP_ COUNTR**	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100 + 0x04*x, (x= 0 to.19)	**TAMP_BKPxR**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

32 Inter-integrated circuit (I2C) interface

32.1 Introduction

The I^2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I^2C bus. It provides multimaster capability, and controls all I^2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus® (power management bus) compatible.

DMA can be used to reduce CPU overload.

32.2 I2C main features

- I^2C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available, depending on the product implementation (see *Section 32.3: I2C implementation*):

- SMBus specification rev 3.0 compatibility:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming

- Wakeup from Stop mode on address match.

32.3 I2C implementation

The devices incorporate up to three I²C-bus controllers, I2C1, I2C2, and I2C3, with full or limited feature sets as shown in the following table.

Table 210. STM32WLEx I2C implementation

I2C features(1)	I2C1(2)	I2C2(2)	I2C3
7-bit addressing mode	X	X	X
10-bit addressing mode	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X
Independent clock	X	X	X
Wakeup from Stop mode	X(3)	X(3)	X(4)
SMBus/PMBus	X	X	X

1. X = supported.
2. The register content is lost in Stop 2 mode.
3. Wakeup supported from Stop 0 and Stop 1 modes.
4. Wakeup supported from Stop 0, Stop 1 and Stop 2 modes.

32.4 I2C functional description

In addition to receiving and transmitting data, this interface converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I^2C bus.

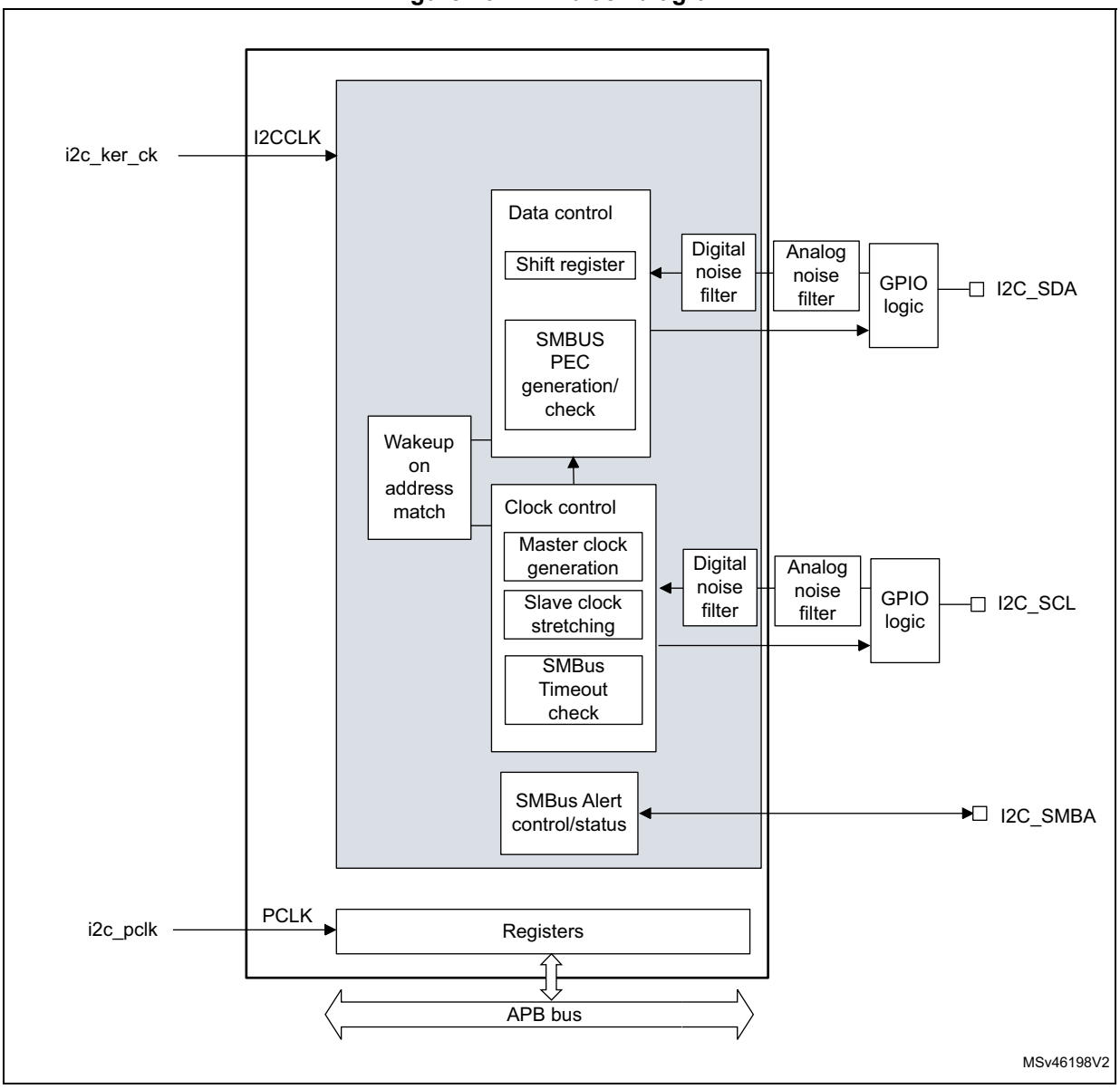
This interface can also be connected to an SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported, the additional optional SMBus Alert pin (SMBA) is also available.

32.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 262*.

Figure 262. I2C block diagram



The I2C is clocked by an independent clock source, which allows the I2C to operate independently from the PCLK frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG). *Refer to Section 32.3: I2C implementation.*

32.4.2 I2C pins and internal signals

Table 211. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I2C data
I2C_SCL	Bidirectional	I2C clock
I2C_SMBA	Bidirectional	SMBus alert

Table 212. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to *Table 225* for the full list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

32.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$ and $t_{I2CCLK} < t_{HIGH}$

with:

t_{LOW}: SCL low time and t_{HIGH}: SCL high time

$t_{filters}$: when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is DNF x t_{I2CCLK}.

The PCLK clock period t_{PCLK} must respect the following condition:

$t_{PCLK} < 4 / 3\ t_{SCL}$

with t_{SCL}: SCL period

Caution: When the I2C kernel is clocked by PCLK, this clock must respect the conditions for t_{I2CCLK}.

32.4.4 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

Communication flow

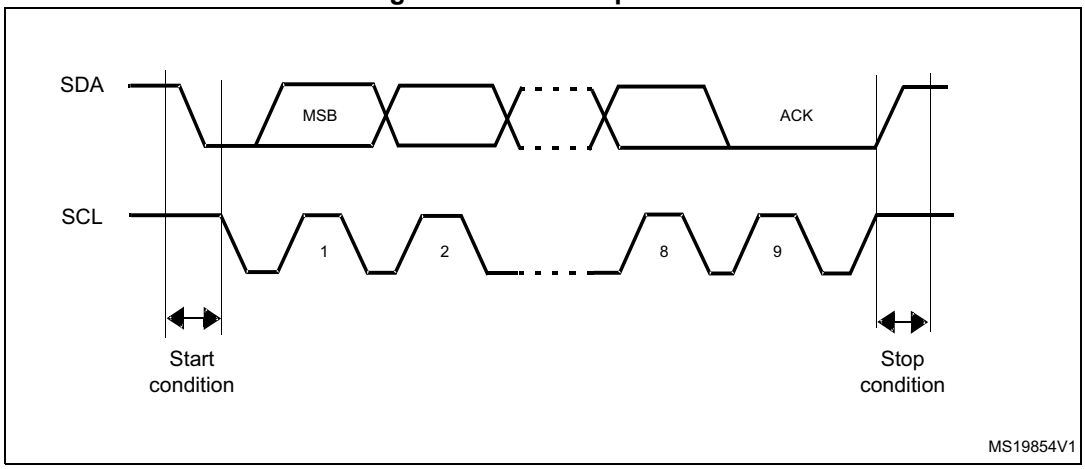
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A ninth clock pulse follows the eight clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter (see *Figure 263*).

Figure 263. I^2C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

32.4.5 I2C initialization

Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller, then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE = 0), the I^2C performs a software reset. Refer to *Section 32.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I^2C specification, which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The

user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows spikes with a programmable length of 1 to 15 I2CCLK periods to be suppressed.

Table 213. Comparison of analog vs. digital filters

-	Analog filter	Digital filter
Pulse width of suppressed spikes	≥ 50 ns	Programmable length from 1 to 15 I2C peripheral clocks
Benefits	Available in Stop mode	– Programmable length: extra filtering capability versus standard requirements – Stable length
Drawbacks	Variation vs. temperature, voltage, process	Wakeup from Stop mode on address match is not available when digital filter is enabled

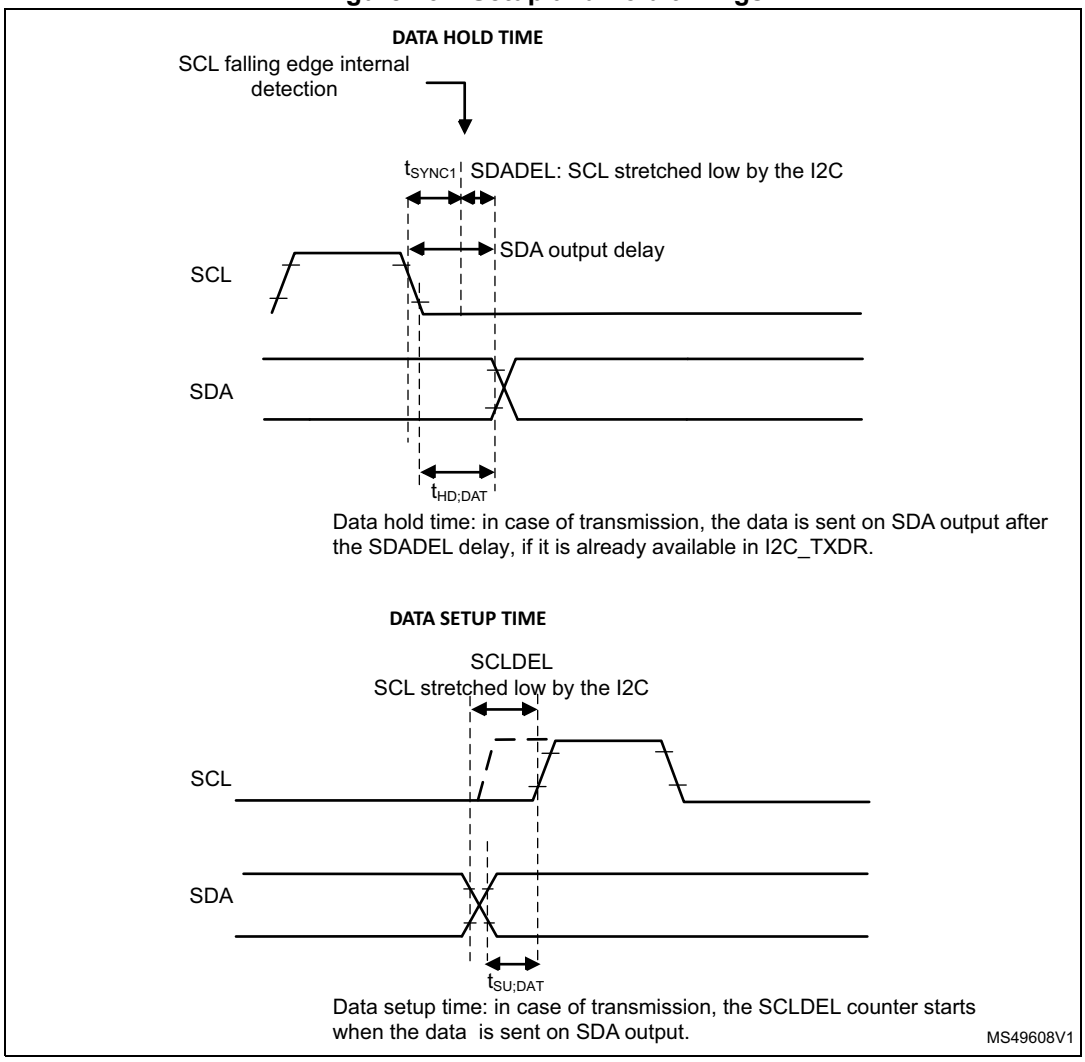
Caution: Changing the filter configuration is not allowed when the I2C is enabled.

I2C timings

The timings must be configured in order to guarantee correct data hold and setup times, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window.

Figure 264. Setup and hold timings



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  T_{SDADEL} impacts the hold time $t_{HD;DAT}$.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC+1) + 1] x t_{I2CCLK} }

t_{SYNC1} duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter: $t_{AF(min)} < t_{AF} < t_{AF(max)}$
- When enabled, input delay brought by the digital filter: t_{DNF} = DNF x t_{I2CCLK}
- Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

$\{t_{f\ (max)} + t_{HD;DAT\ (min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: *$t_{AF(min)}$ / $t_{AF(max)}$ are part of the equation only when the analog filter is enabled. Refer to device datasheet for t_{AF} values.*

The maximum $t_{HD;DAT}$ can be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - 260\ ns - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$.

Note: *This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.*

Refer to *Table 214* for t_f, t_r, $t_{HD;DAT}$ and $t_{VD;DAT}$ standard values.

- After t_{SDADEL} delay, or after sending SDA output in case the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, SCL line is kept at low level during the setup time. This setup time is t_{SCLDEL} = (SCLDEL+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC+1)] \times t_{I2CCLK}]\} - 1 \leq SCLDEL$

Refer to *Table 214* for t_r and $t_{SU;DAT}$ standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

Note: *At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both*

transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOSTRETCH = 1 in slave mode, the SCL is not stretched. Consequently the SDADEL must be programmed in such a way to guarantee also a sufficient setup time.

Table 214. I^2C-SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output.
  This delay is $t_{SCLL} = (SCLL + 1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$.
  t_{SCLL} impacts the SCL low time t_{LOW}.
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is $t_{SCLH} = (SCLH + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$. t_{SCLH} impacts the SCL high time t_{HIGH}.

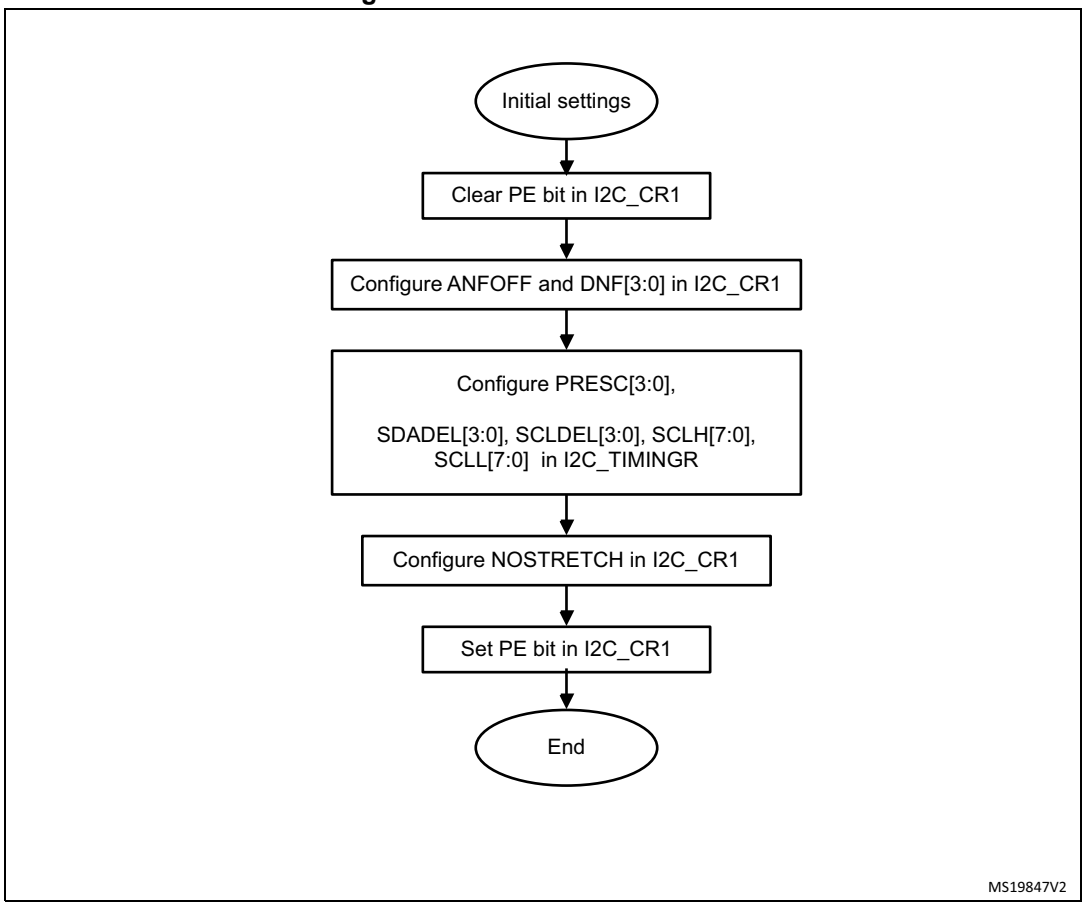
Refer to *I2C master initialization* for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to *I2C slave initialization* for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 265. I2C initialization flow



32.4.6 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least three APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

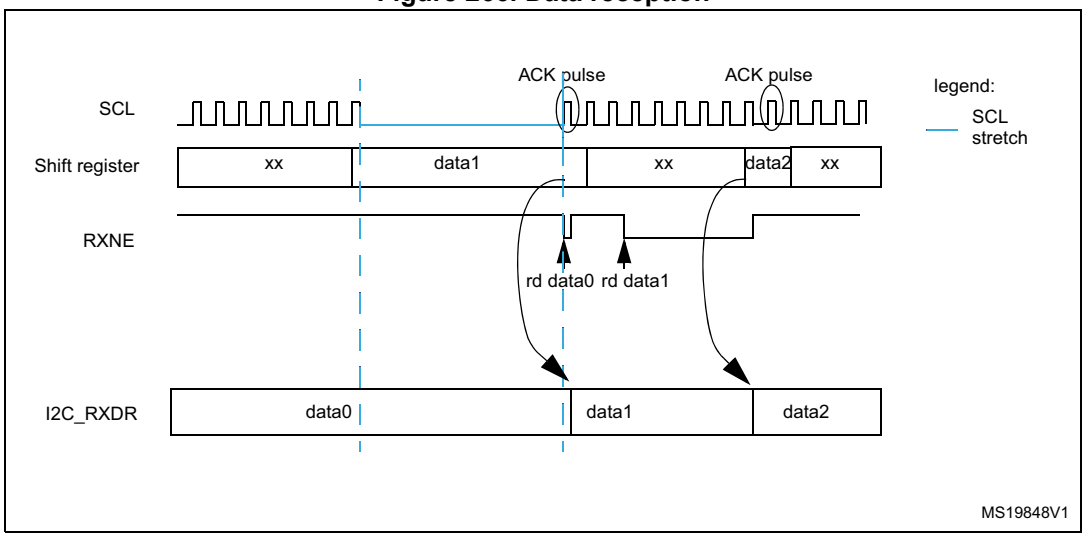
32.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the eighth and ninth SCL pulse (before the acknowledge pulse).

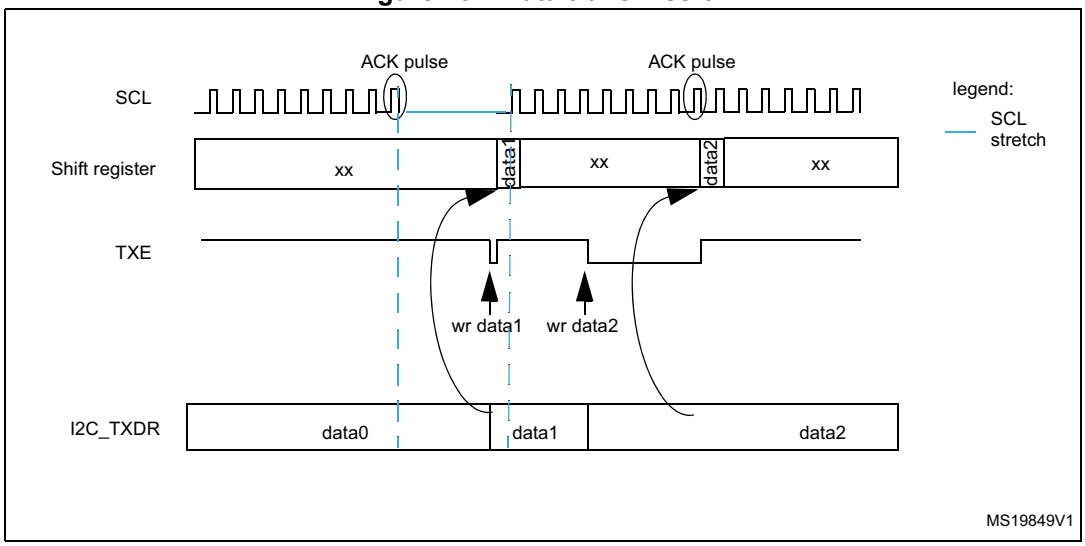
Figure 266. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the ninth SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE = 1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the ninth SCL pulse.

Figure 267. Data transmission



Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (slave byte control) bit in the I2C_CR1 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES is transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in master mode, the counter can be used in two modes:

- **Automatic end mode** (AUTOEND = ‘1’ in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field is transferred.
- **Software end mode** (AUTOEND = ‘0’ in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the master wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 215. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Master Tx/Rx NBYTES + STOP	x	0	1
Master Tx/Rx + NBYTES + RESTART	x	0	0
Slave Tx/Rx all received bytes ACKed	0	x	x
Slave Rx with ACK control	1	1	x

32.4.8 I2C slave mode

I2C slave initialization

In order to work in slave mode, the user must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.

  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.
- If additional slave addresses are required, the second slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.
- The general call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE = 1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}.

Slave without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Slave byte control mode

In order to allow byte ACK control in slave reception mode, The Slave byte control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

The Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD = 1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and ninth SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

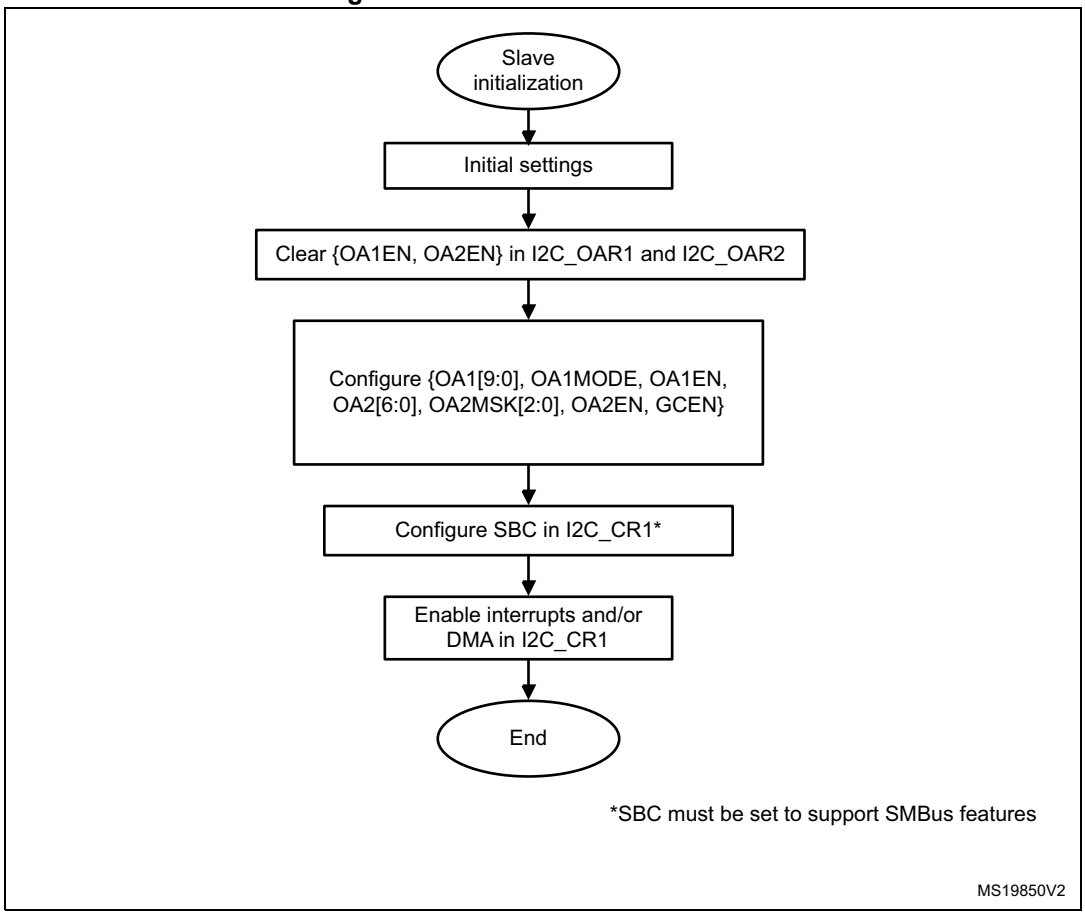
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The Slave byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 268. Slave initialization flow



Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to ‘0’. In this case, If TXE = 0 when the slave address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave byte control mode (SBC = 1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event is needed, (transmit interrupt or transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

Figure 269. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0

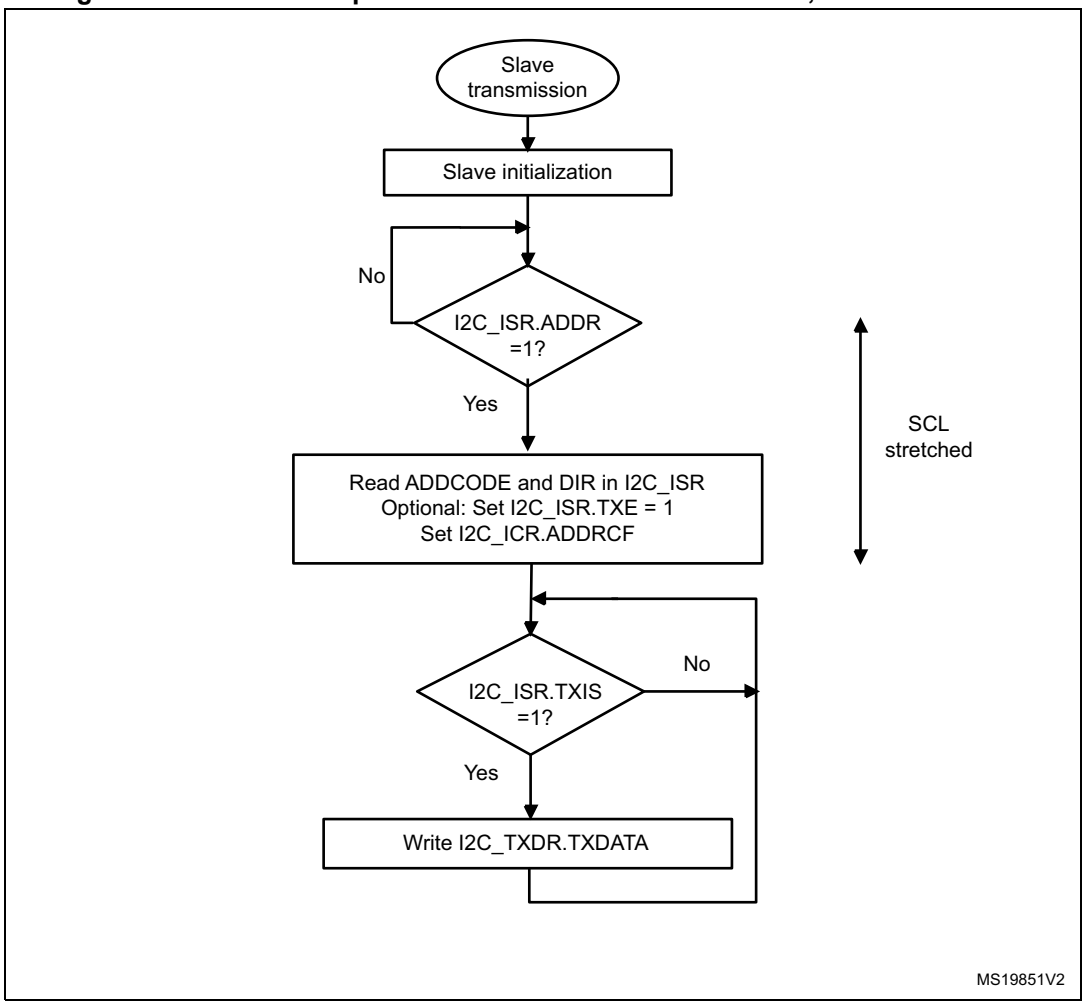


Figure 270. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1

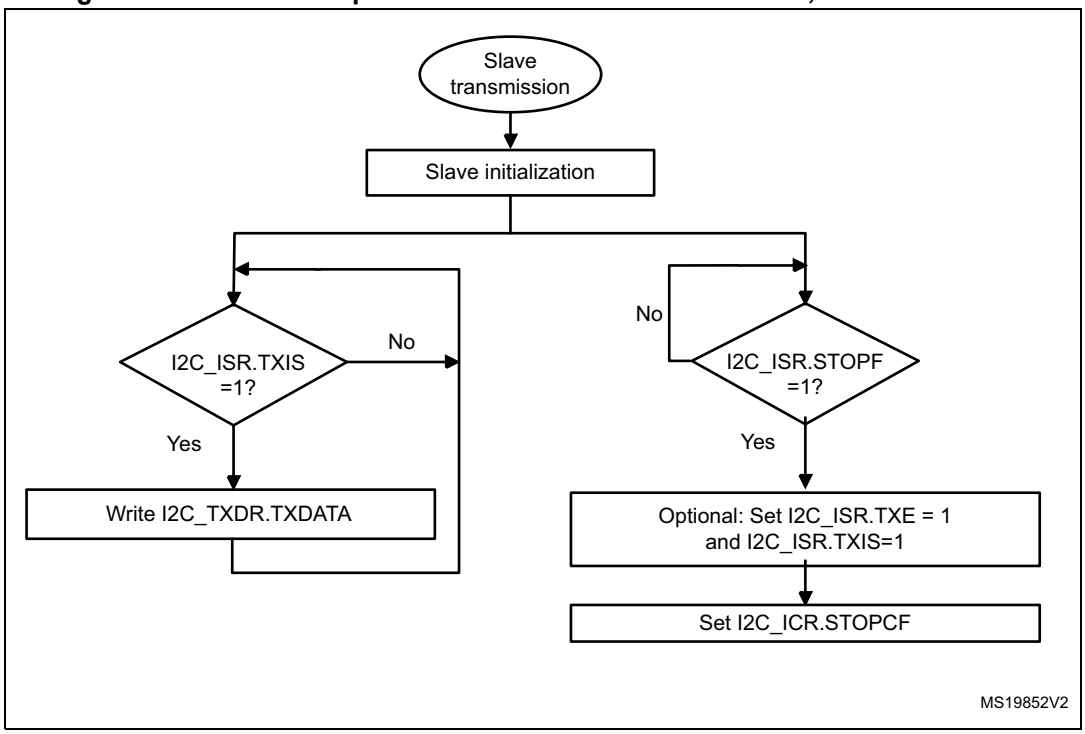
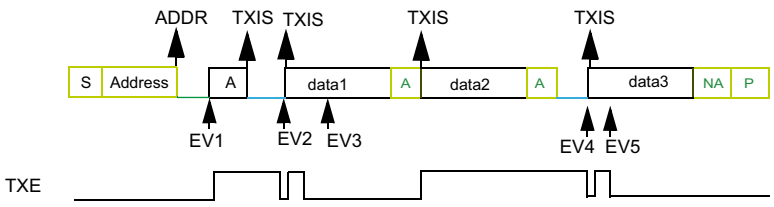


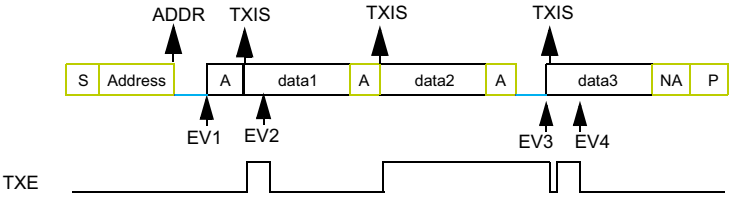
Figure 271. Transfer bus diagrams for I2C slave transmitter

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



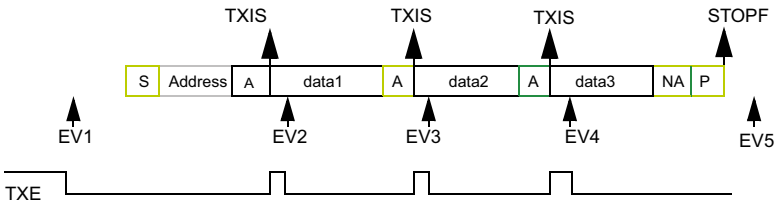
EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:



EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MS19853V2

Slave receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 272. Transfer sequence flow for slave receiver with NOSTRETCH = 0

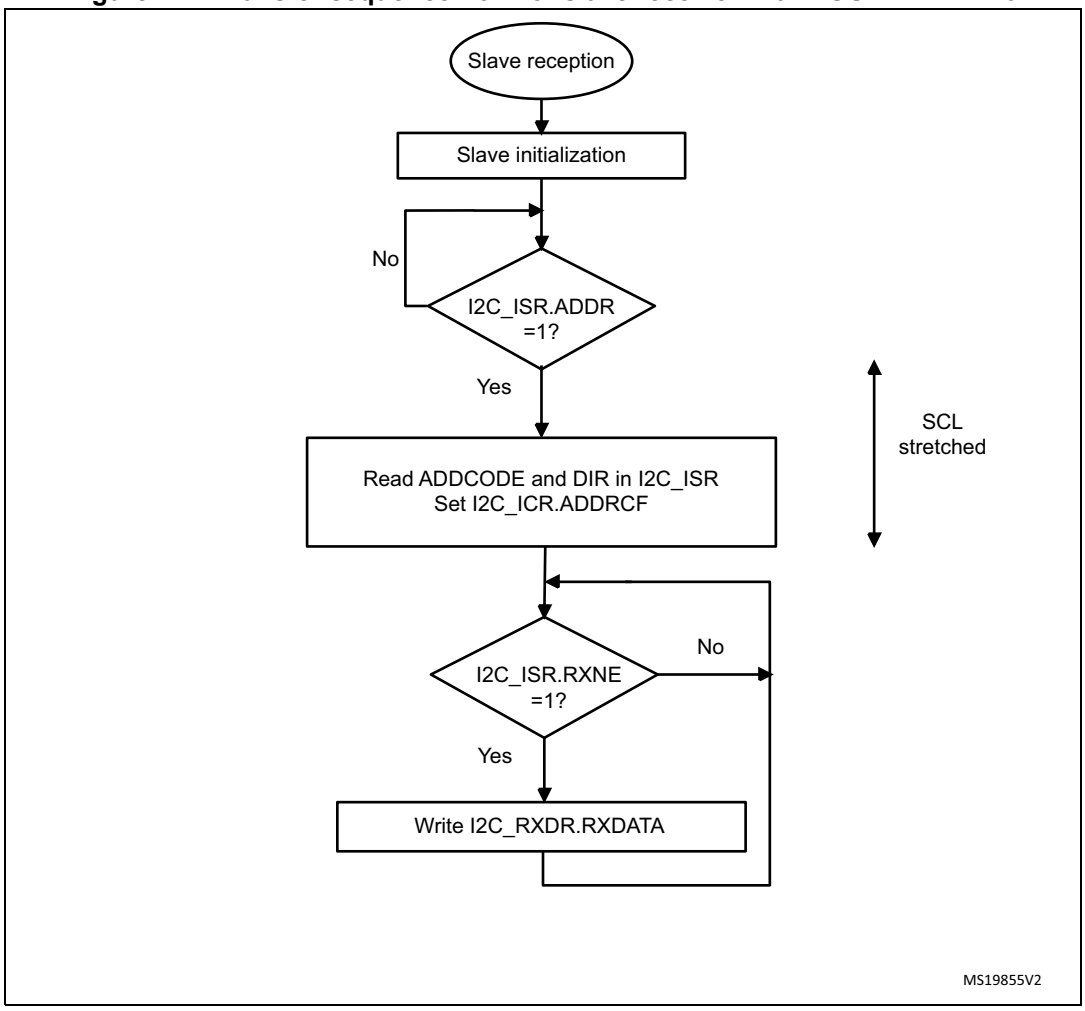


Figure 273. Transfer sequence flow for slave receiver with NOSTRETCH = 1

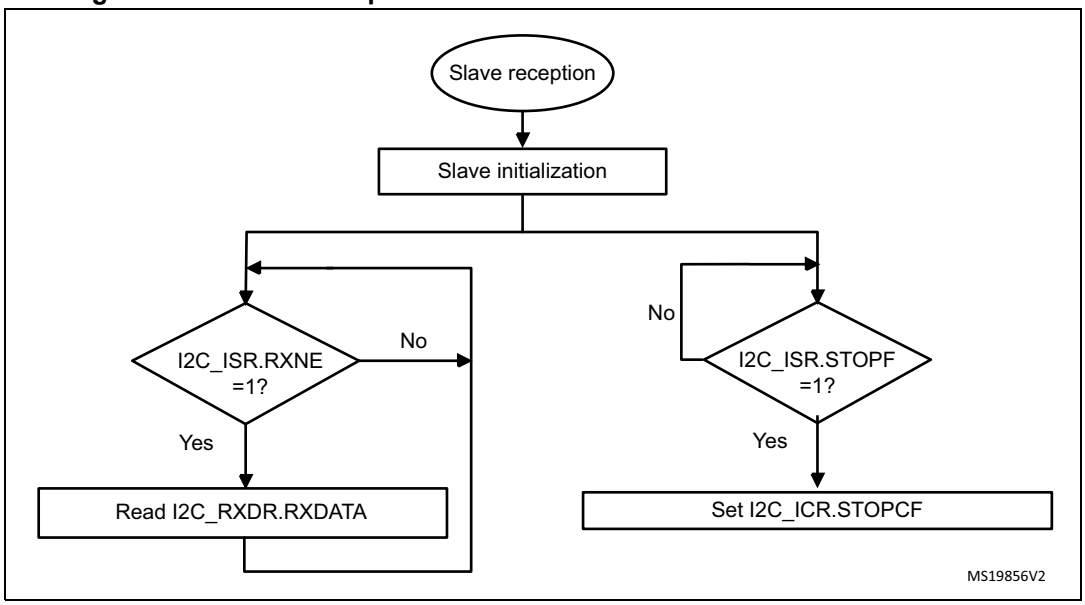
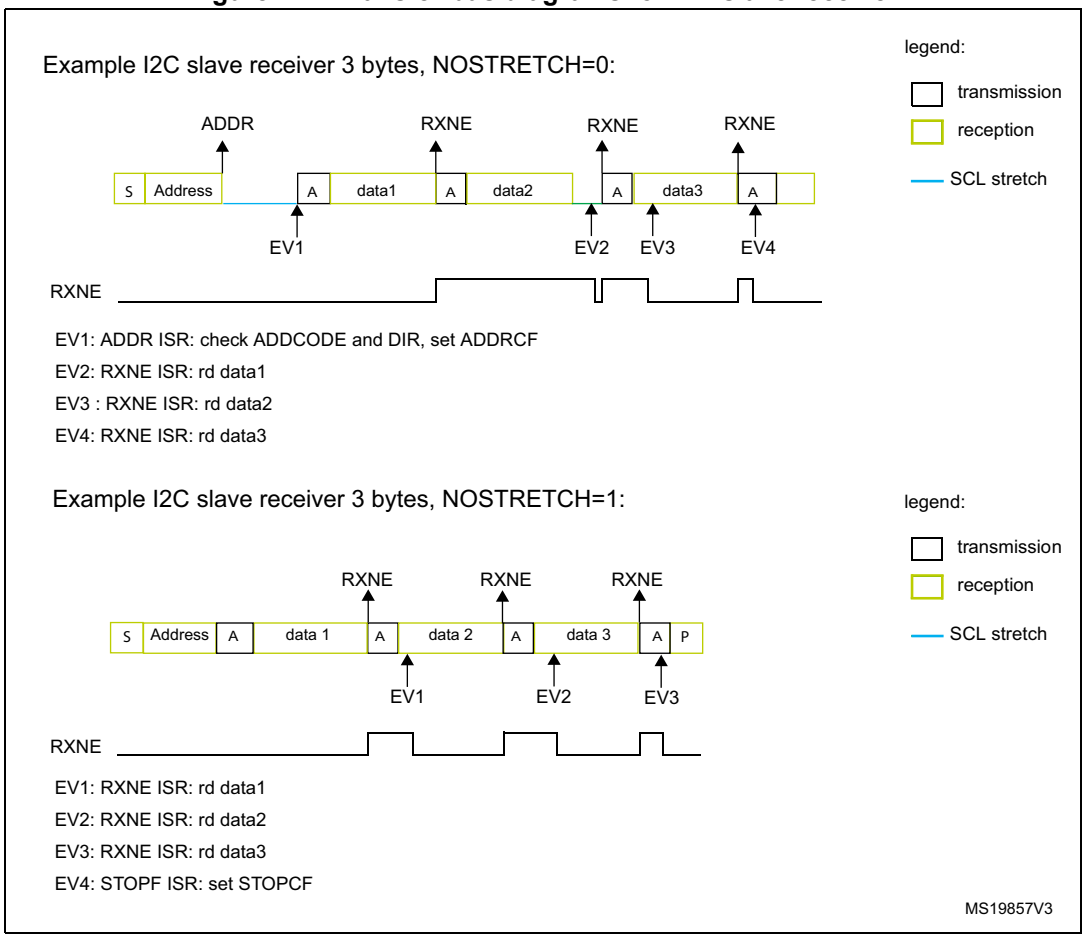


Figure 274. Transfer bus diagrams for I2C slave receiver



32.4.9 I2C master mode

I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the master clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$

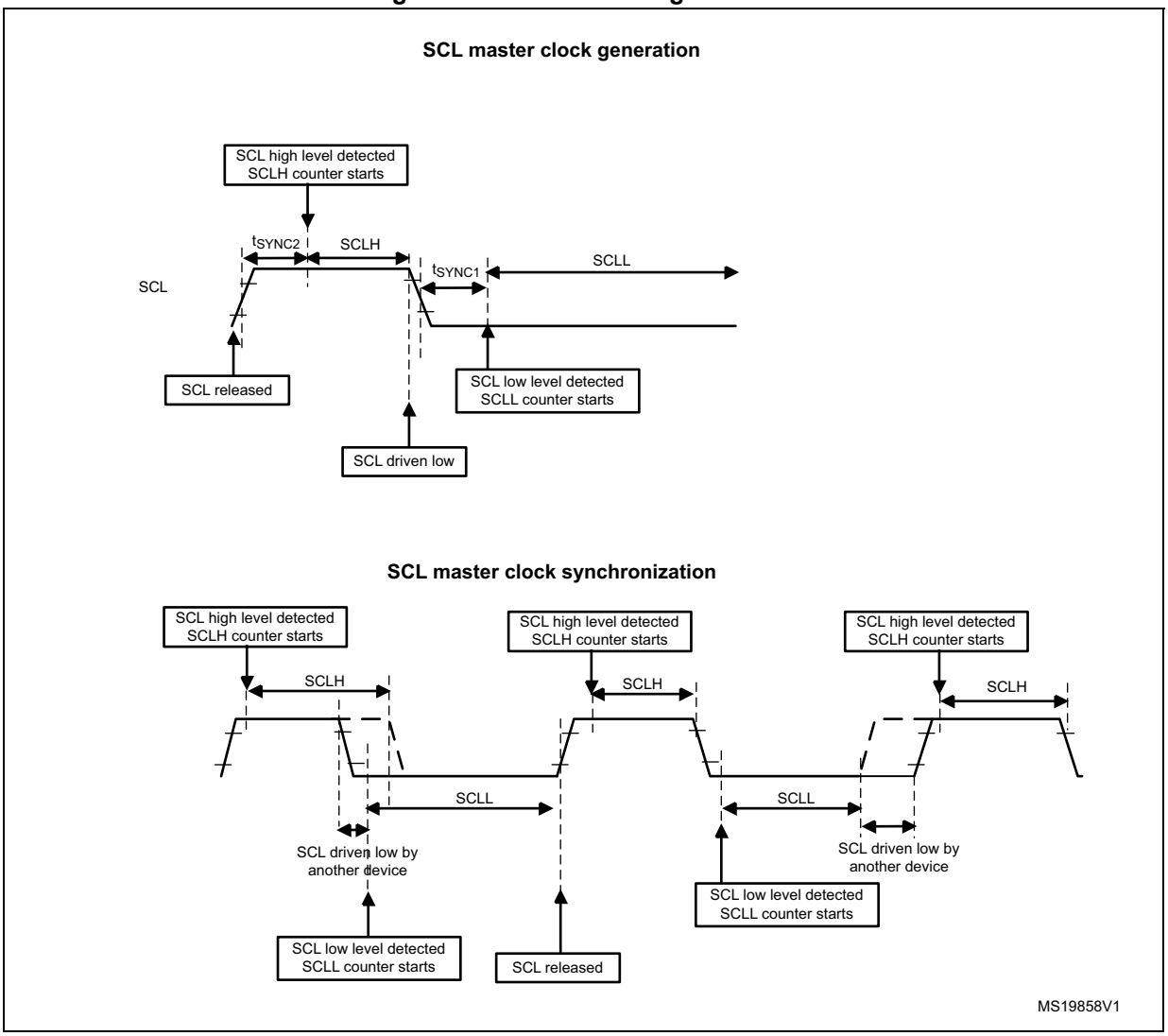
The duration of t_{SYNC1} depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of t_{SYNC2} depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

Figure 275. Master clock generation



Caution: To be I^2C or SMBus compliant, the master clock must respect the timings given in the table below.

Table 216. I^2C-SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: SCLL is also used to generate the t_{BUF} and $t_{SU:STA}$ timings.

SCLH is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 32.4.10: I2C_TIMINGR register configuration examples* for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

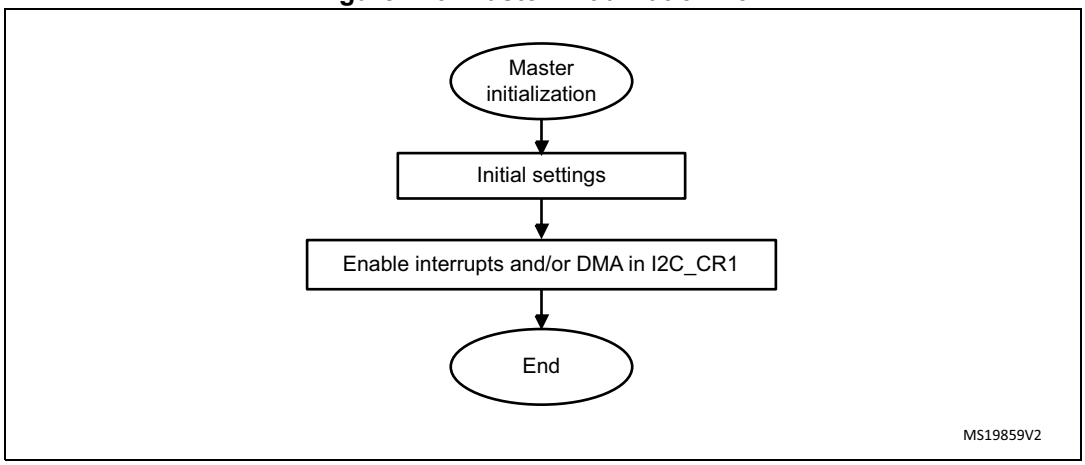
Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t_{BUF}.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

Note: The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.
In 10-bit addressing mode, when the Slave Address first 7 bits are NACKed by the slave, the master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, in order to stop sending the slave address.
If the I2C is addressed as a slave (ADDR = 1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared.

Note: The same procedure is applied for a Repeated start condition. In this case BUSY = 1.

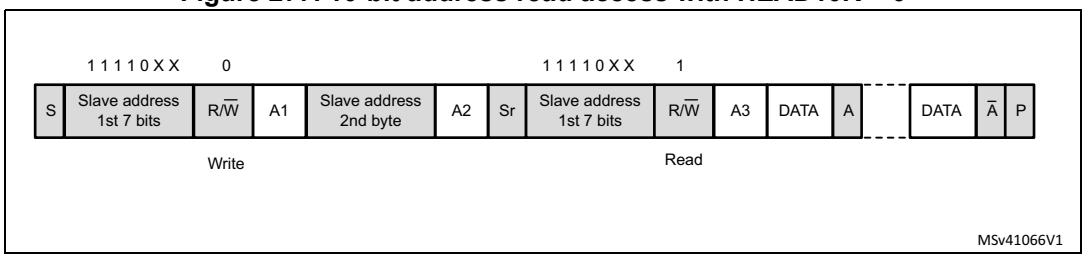
Figure 276. Master initialization flow



Initialization of a master receiver addressing a 10-bit address slave

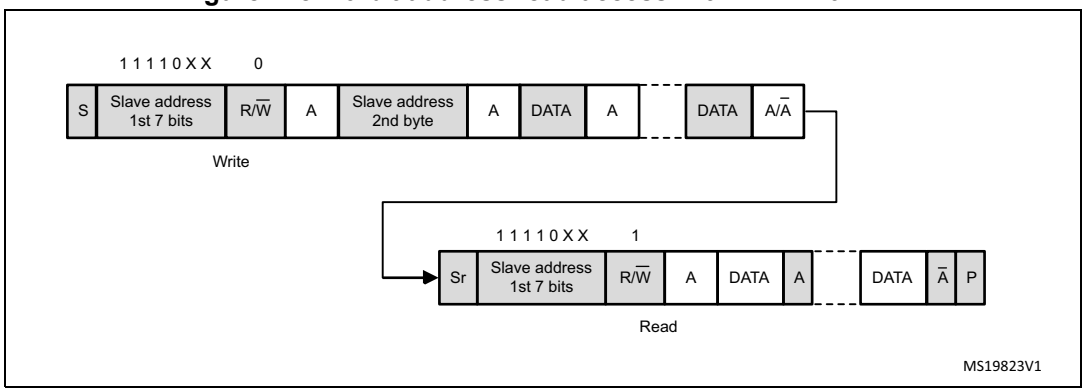
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address second byte + REStart + Slave address 10-bit header Read

Figure 277. 10-bit address read access with HEAD10R = 0



- If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10 bit slave address configured with HEAD10R = 1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

Figure 278. 10-bit address read access with HEAD10R = 1



Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.

Figure 279. Transfer sequence flow for I2C master transmitter for N≤255 bytes

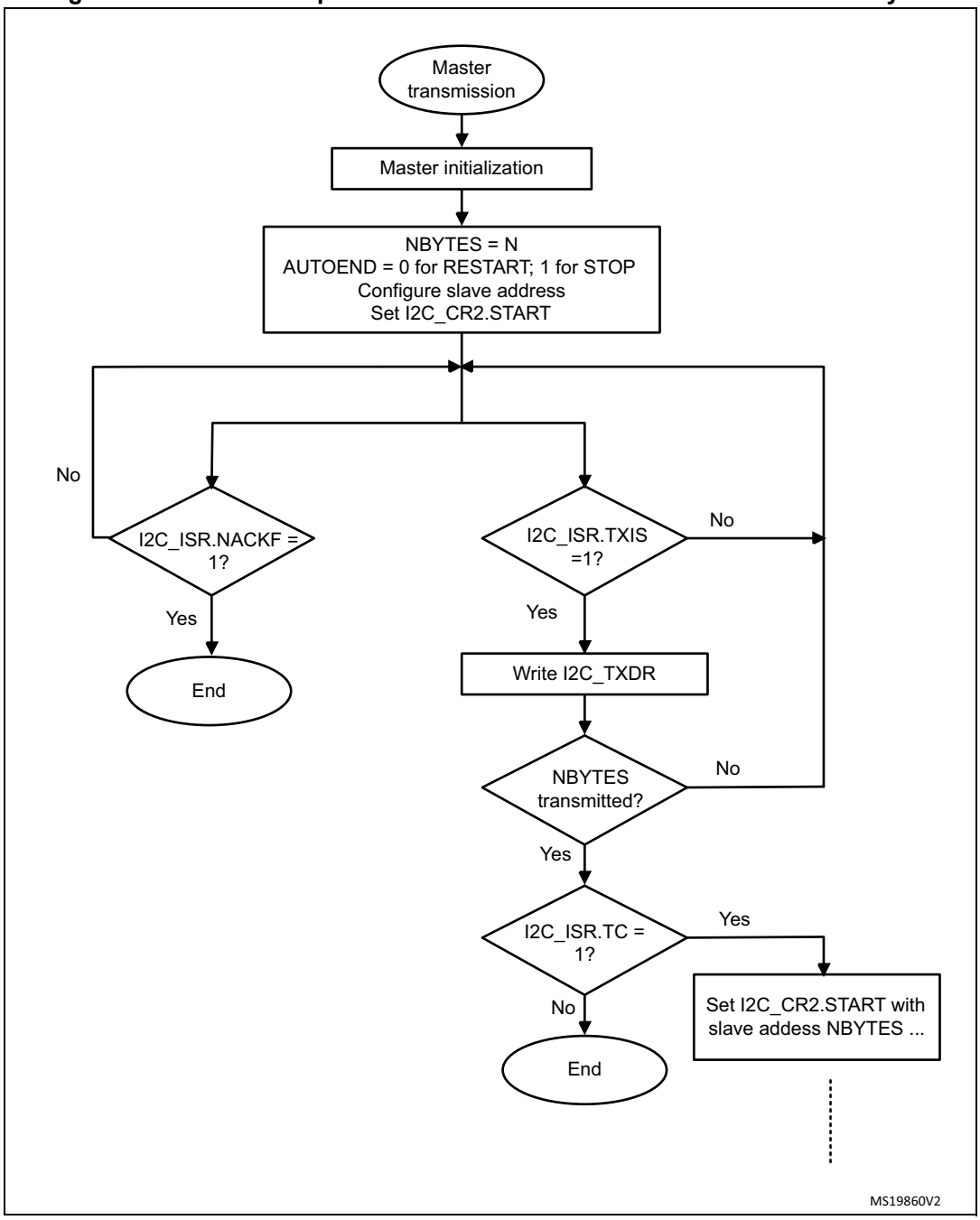


Figure 280. Transfer sequence flow for I2C master transmitter for N>255 bytes

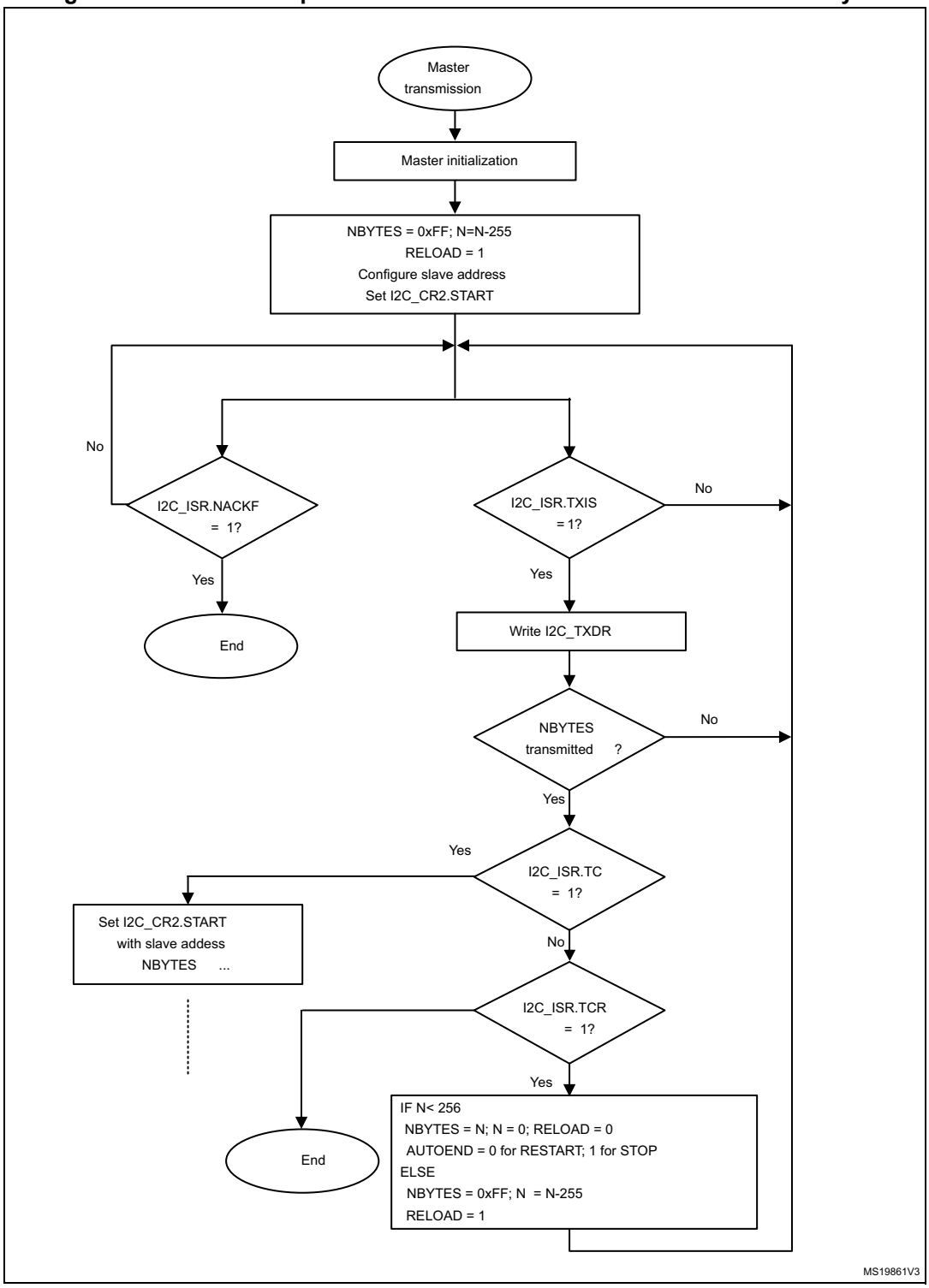
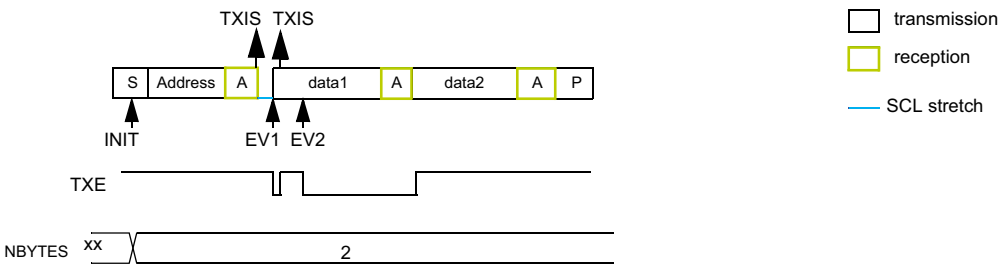


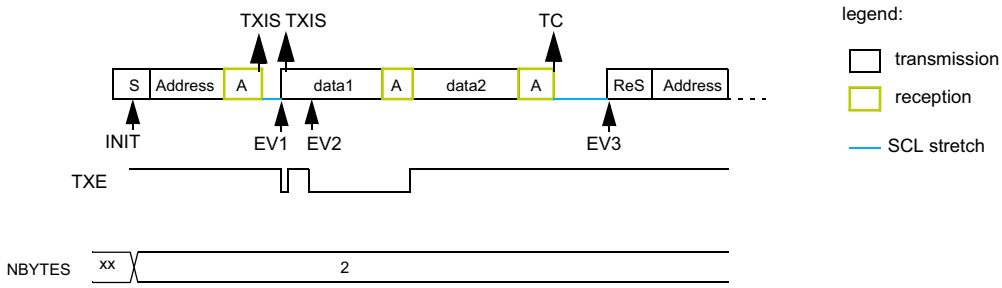
Figure 281. Transfer bus diagrams for I2C master transmitter

Example I2C master transmitter 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19862V2

Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 282. Transfer sequence flow for I2C master receiver for N≤255 bytes

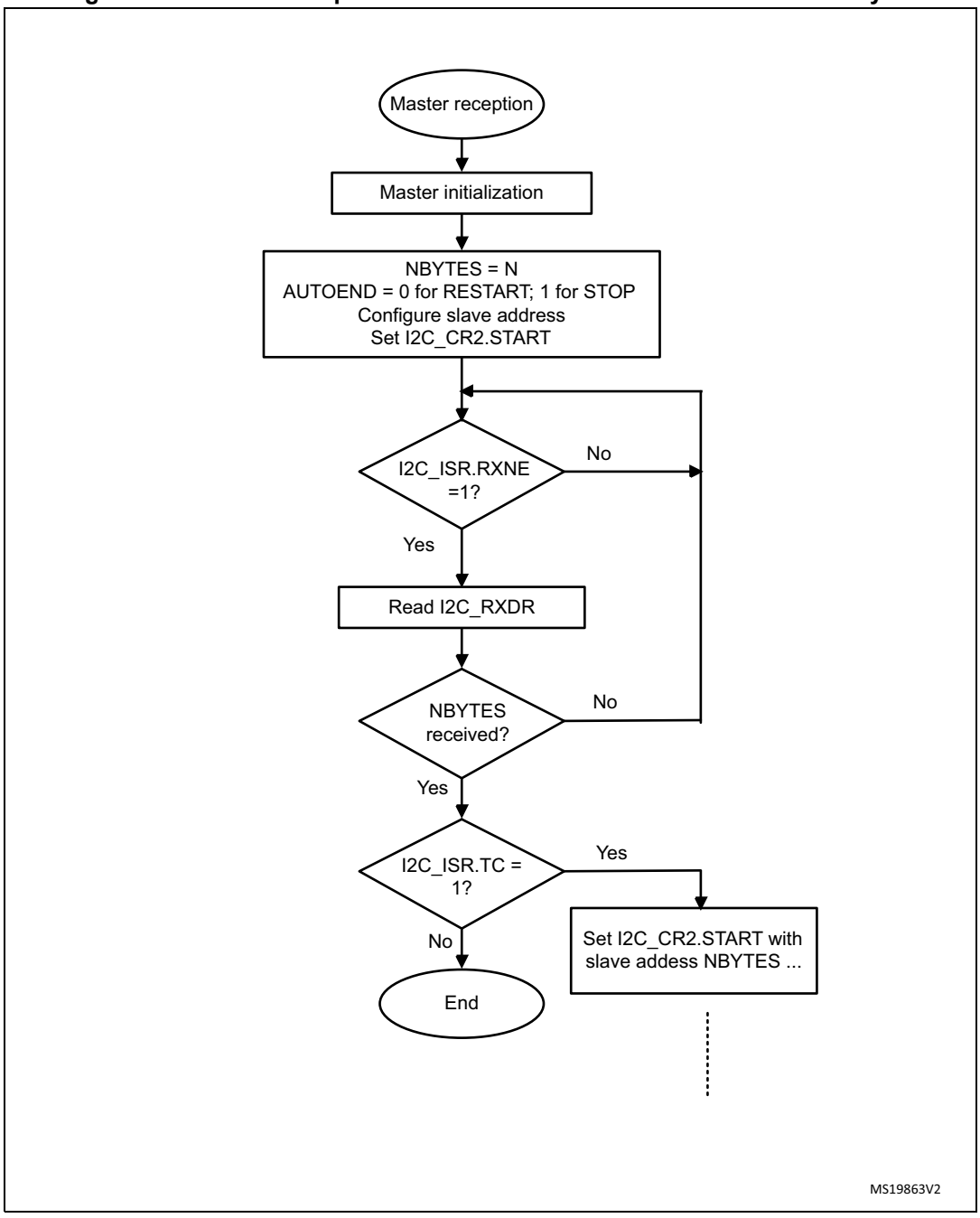


Figure 283. Transfer sequence flow for I2C master receiver for N >255 bytes

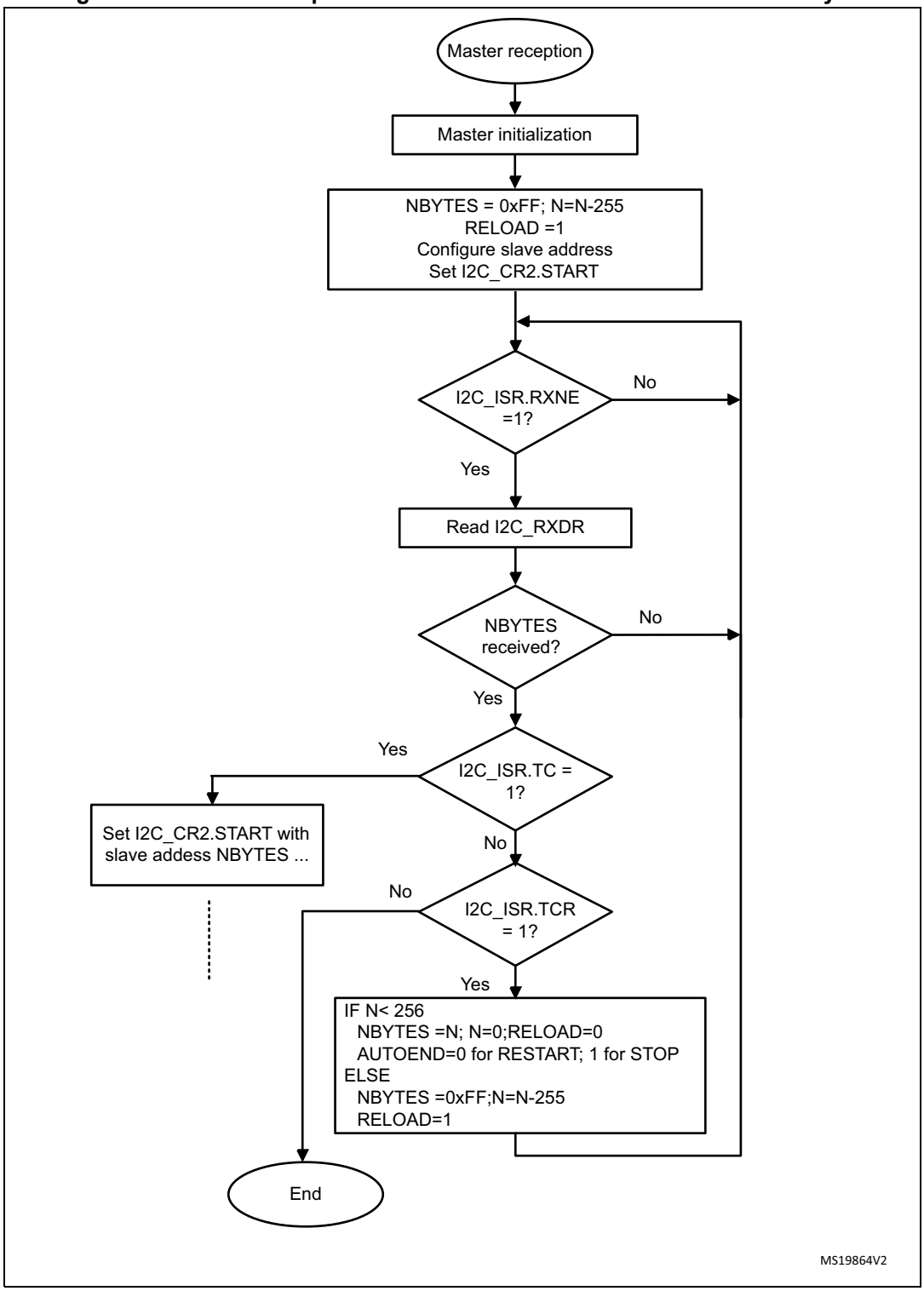
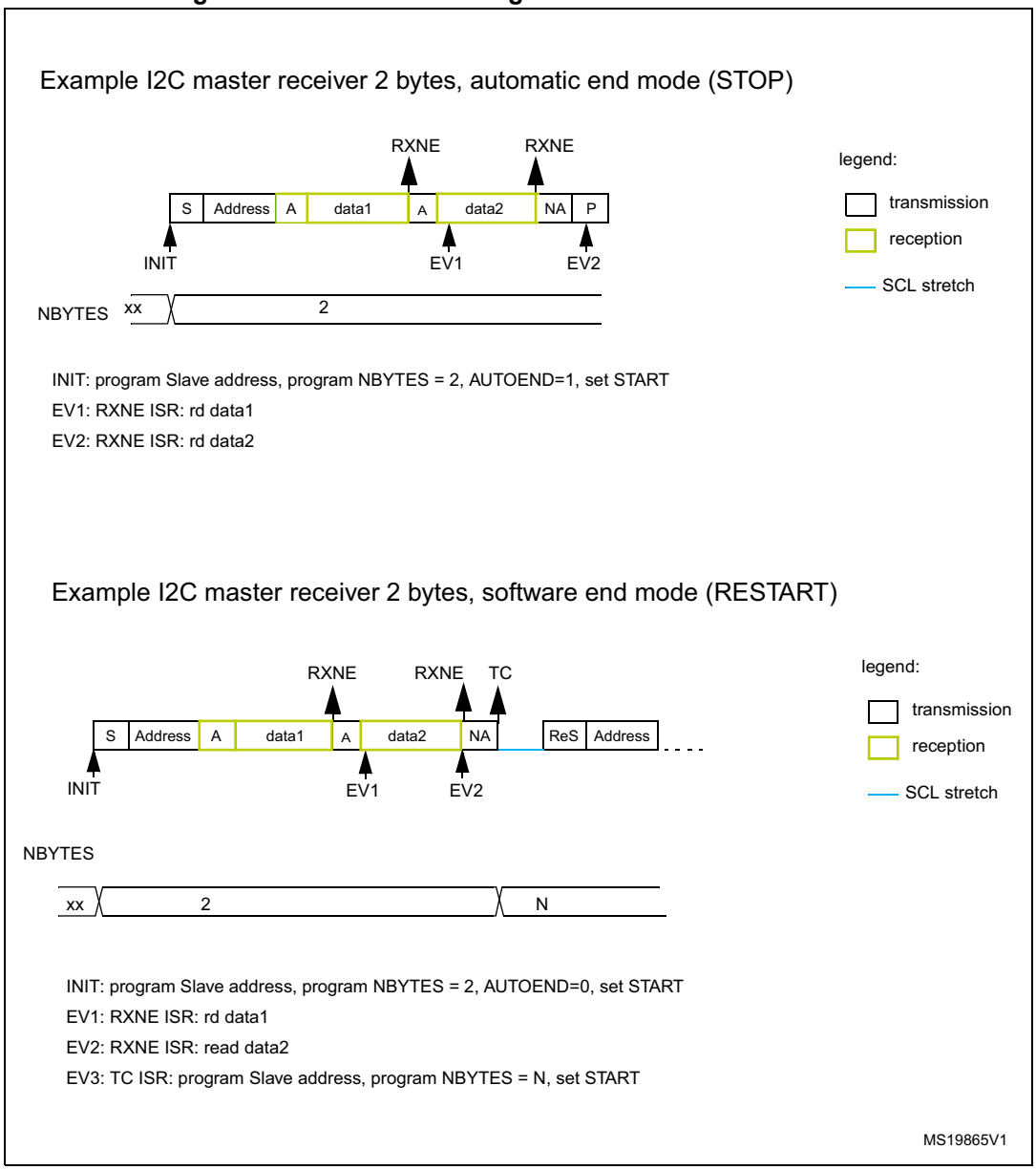


Figure 284. Transfer bus diagrams for I2C master receiver



32.4.10 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I^2C specification. In order to get more accurate configuration values, the STM32CubeMX tool (I2C Configuration window) must be used.

Table 217. Examples of timing settings for f_{I2CCLK} = 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~2000 ns(4)
SDADEL	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 218. Examples of timings settings for f_{I2CCLK} = 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~1000 ns(4)
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.

2. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x t_{I2CCLK} = 250 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 1000 ns.
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x t_{I2CCLK} = 250 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 750 ns.
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x t_{I2CCLK} = 250 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 500 ns.

32.4.11 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to *Section 32.3: I2C implementation*.

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. The SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBus specification (http://smbus.org).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification (http://smbus.org).

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus address resolution protocol, refer to SMBus specification (http://smbus.org).

Received command and data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to *Slave byte control mode* for more details.

Host notify protocol

This peripheral supports the host notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the alert response address.

When configured as a slave device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0*.*

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x_8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

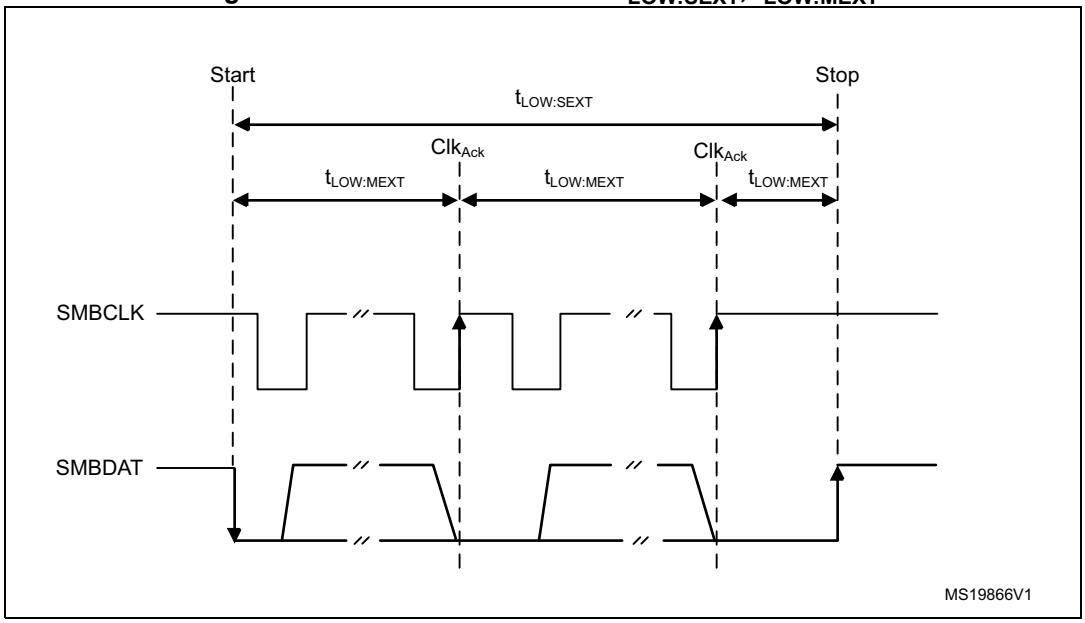
This peripheral embeds hardware timers in order to be compliant with the three timeouts defined in SMBus specification.

Table 219. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (slave device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (master device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2. $t_{LOW:MEXT}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

Figure 285. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for t_{IDLE} greater than $t_{HIGH,MAX}$. (refer to *Table 214*)

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

32.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to *Section 32.3: I2C implementation*.

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

Received command and data acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave byte control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to *Slave byte control mode on page 961* for more details.

Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to *Bus idle detection on page 984* for more details.

- The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The alert response address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES - 1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C is enabled.

Table 220. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Master Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
Master Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
Slave Tx/Rx with PEC	1	0	x	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- $t_{TIMEOUT}$ check

  In order to enable the $t_{TIMEOUT}$ check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the $t_{TIMEOUT}$ parameter. The TIDLE bit must be configured to ‘0’ in order to detect the SCL low level timeout.

  Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register.

  If SCL is tied low for a time greater than (TIMEOUTA+1) x 2048 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 221*.

Caution: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

  Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check $t_{LOW:SEXT}$ for a slave and $t_{LOW:MEXT}$ for a master. As the standard specifies only a maximum, the user can choose the same value for the both.

  Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

  If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TIMEOUTB+1) x 2048 x t_{I2CCLK}, and in the timeout interval described in *Bus idle detection* section, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 222*

Caution: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

Bus idle detection

In order to enable the t_{IDLE} check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the t_{IDLE} parameter. The TIDLE bit must be configured to ‘1 in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than (TIMEOUTA+1) x 4 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

Refer to *Table 223*.

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

32.4.13 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 32.3: I2C implementation*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 221. Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{TIMEOUT}$ = 25 ms)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms

- Configuring the maximum duration of $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ to 8 ms:

Table 222. Examples of TIMEOUTB settings for various I2CCLK frequencies

f_{I2CCLK}	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 µs

Table 223. Examples of TIMEOUTA settings for various I2CCLK frequencies (max t_{IDLE} = 50 µs)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	t_{TIDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs

32.4.14 SMBus slave mode

This section is relevant only when the SMBus feature is supported. Refer to *Section 32.3: I2C implementation*.

In addition to I2C slave transfer management (refer to *Section 32.4.8: I2C slave mode*) some additional software flows are provided to support the SMBus.

SMBus slave transmitter

When the IP is used in SMBus, SBC must be programmed to ‘1’ in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTES - 1 and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES - 1 data transfer.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 286. Transfer sequence flow for SMBus slave transmitter N bytes + PEC

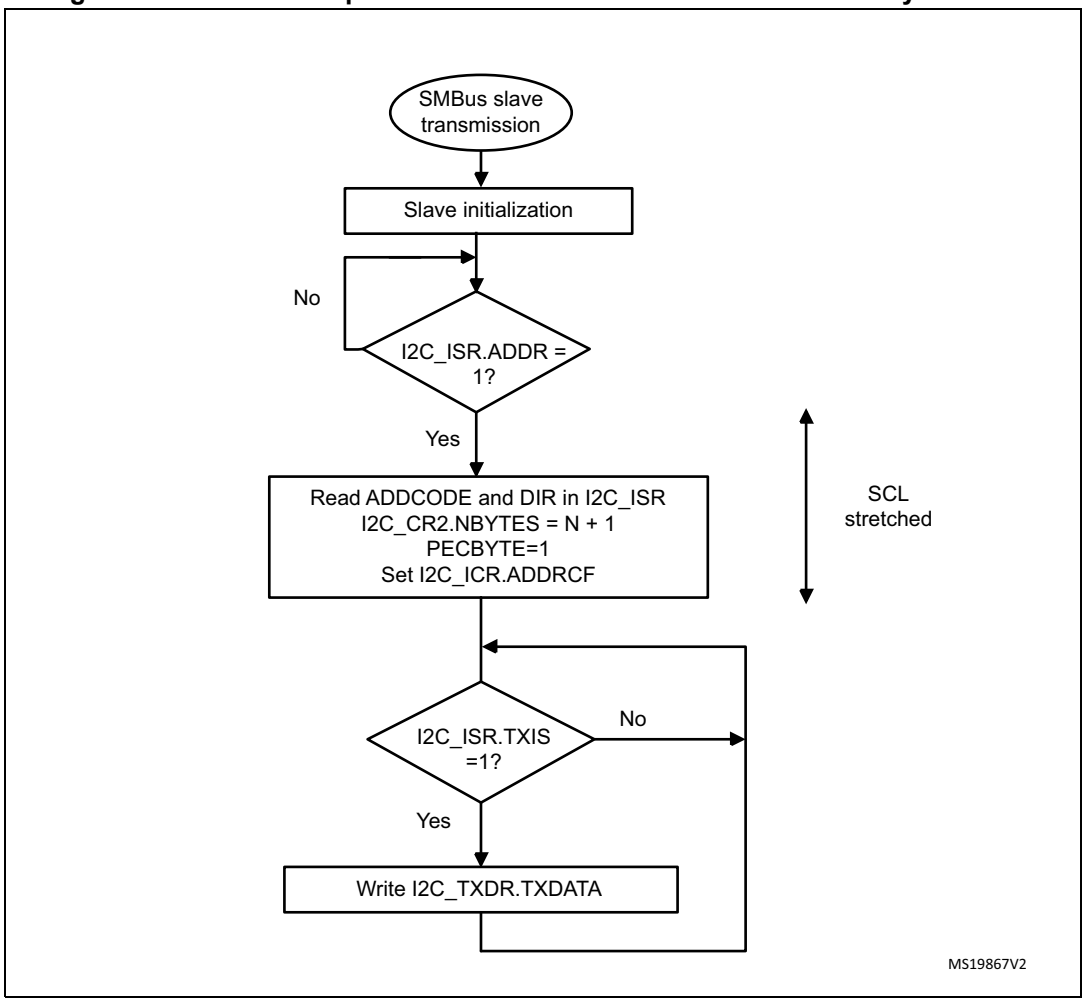
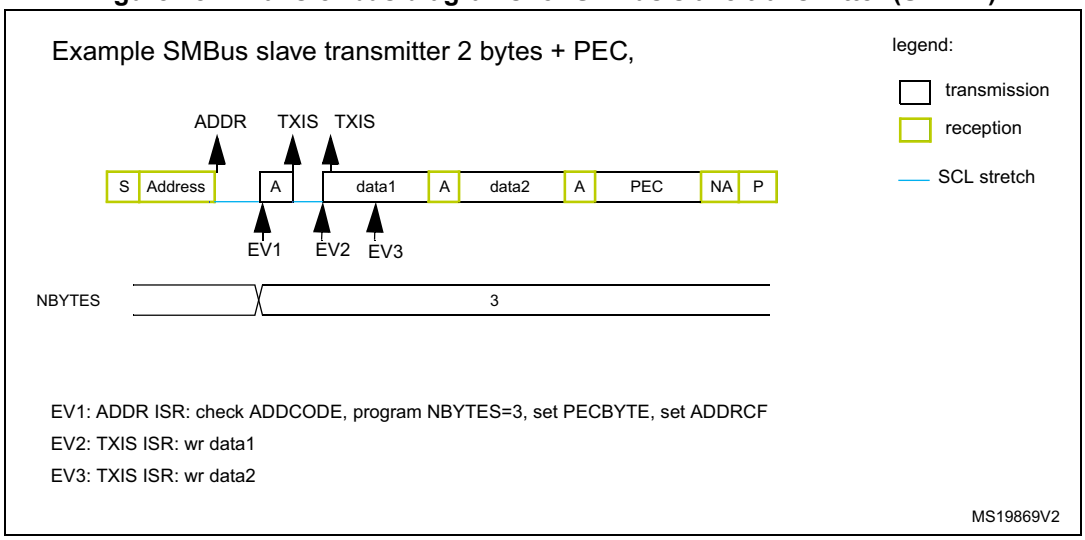


Figure 287. Transfer bus diagrams for SMBus slave transmitter (SBC=1)



SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to ‘1’ in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to *Slave byte control mode* for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES - 1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, the user can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES - 1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 288. Transfer sequence flow for SMBus slave receiver N Bytes + PEC

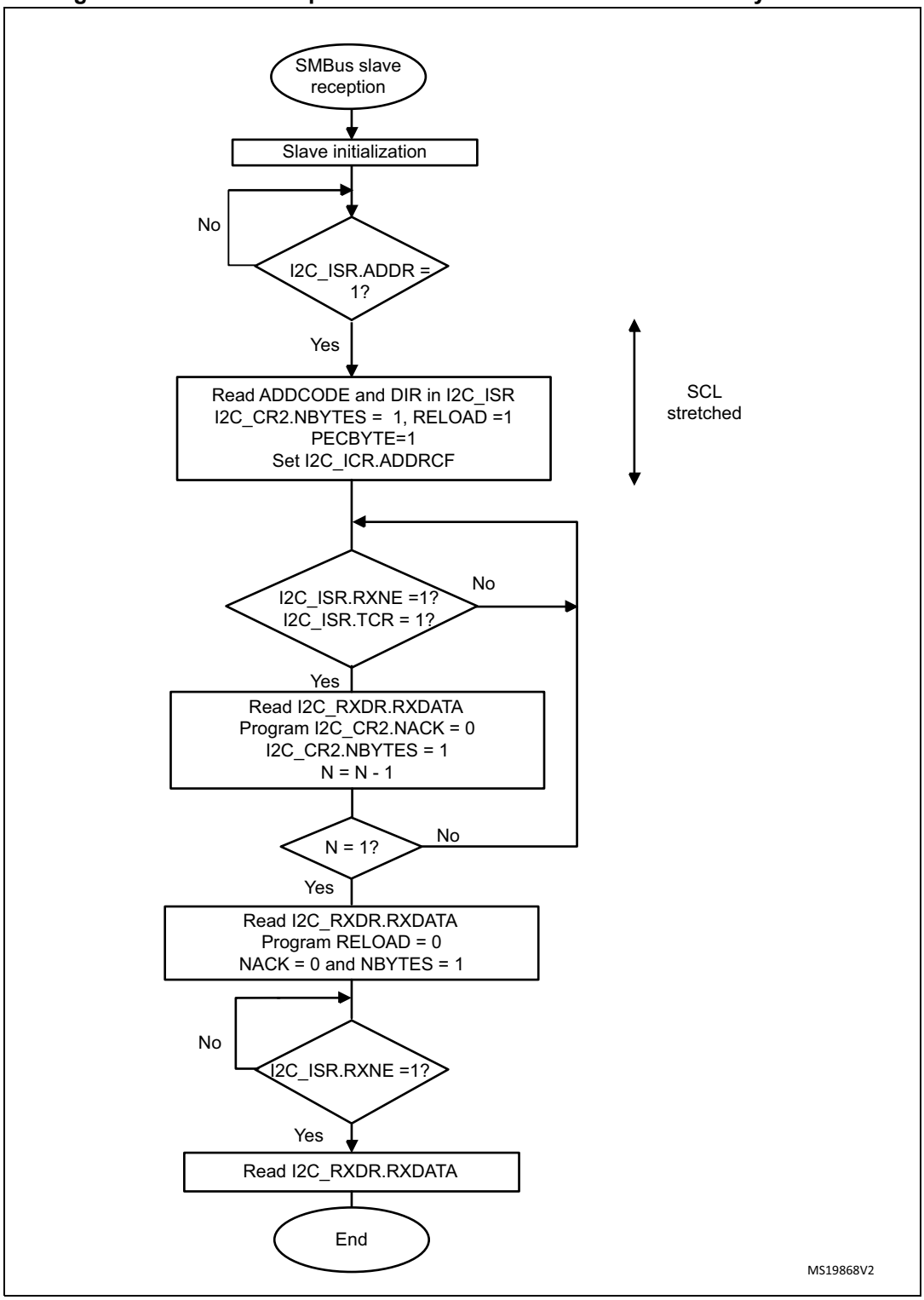
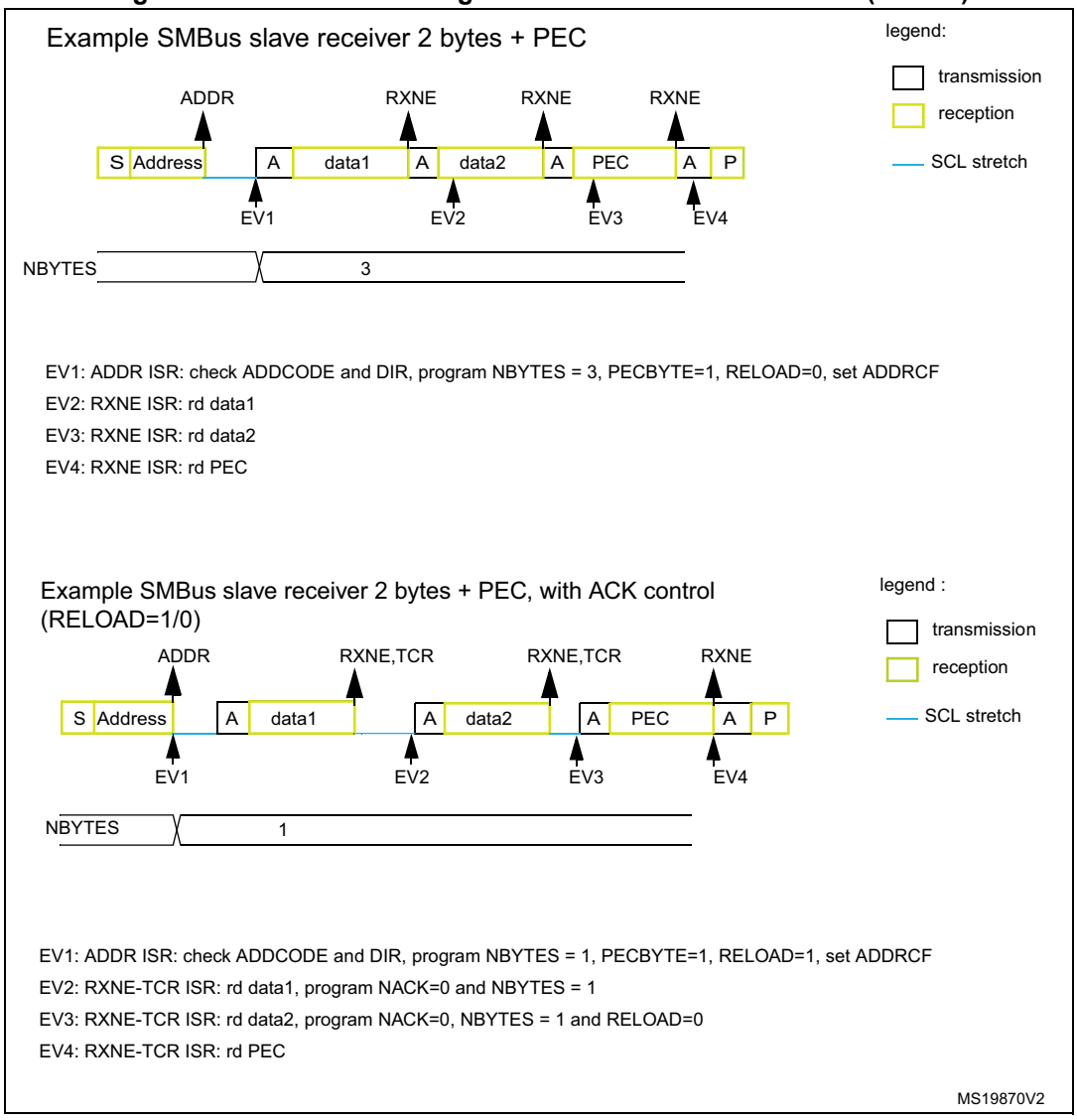


Figure 289. Bus transfer diagrams for SMBus slave receiver (SBC=1)



This section is relevant only when the SMBus feature is supported. Refer to *Section 32.3: I2C implementation*.

In addition to I2C master transfer management (refer to *Section 32.4.9: I2C master mode*), some additional software flows are provided to support the SMBus.

SMBus master transmitter

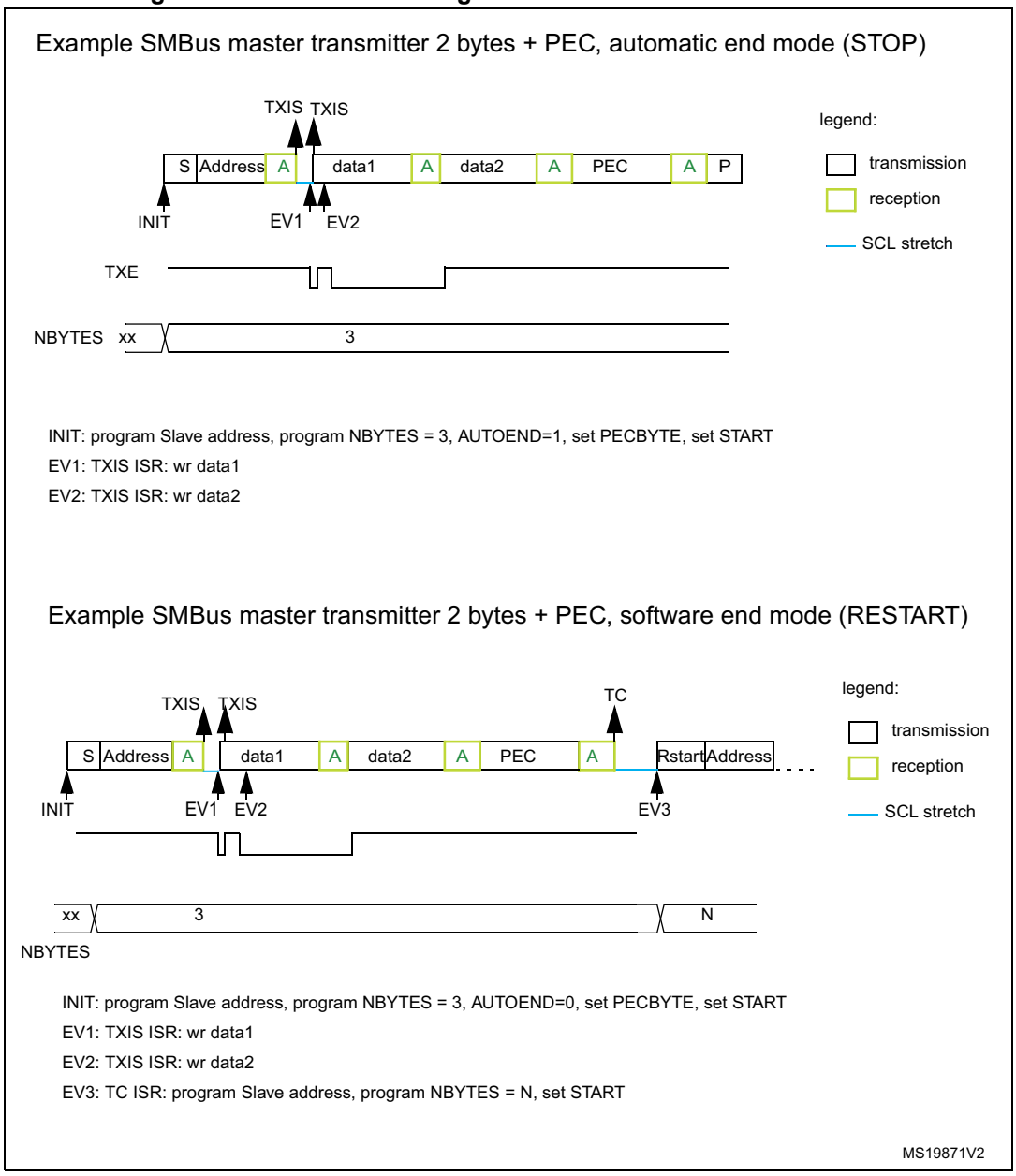
When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES - 1. So if the PECBYTE bit is set when NBYTES = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES - 1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 290. Bus transfer diagrams for SMBus master transmitter



SMBus master receiver

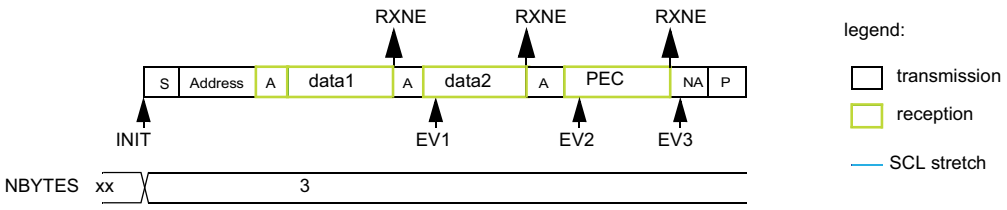
When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES - 1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES - 1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

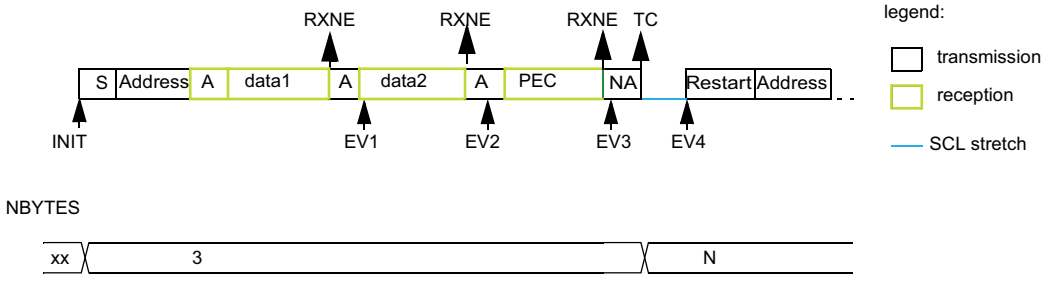
Figure 291. Bus transfer diagrams for SMBus master receiver

Example SMBus master receiver 2 bytes + PEC, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: rd PEC

Example SMBus master receiver 2 bytes + PEC, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: read PEC
EV4: TC ISR: program Slave address, program NBYTES = N, set START

MS19872V2

32.4.15 Wakeup from Stop mode on address match

This section is relevant only when wakeup from Stop mode feature is supported. Refer to *Section 32.3: I2C implementation*.

The I2C is able to wakeup the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wakeup from Stop mode is enabled by setting the WUPEN bit in the I2C_CR1 register. The HSI16 oscillator must be selected as the clock source for I2CCLK in order to allow wakeup from Stop mode.

During Stop mode, the HSI16 is switched off. When a START is detected, the I2C interface switches the HSI16 on, and stretches SCL low until HSI16 is woken up.

HSI16 is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wakeup time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI16 is switched off again and the MCU is not woken up.

Note: If the I2C clock is the system clock, or if WUPEN = 0, the HSI16 *is not switched on after a START is received.*

Only an ADDR interrupt can wakeup the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wakeup from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

Caution: This feature is available only when the I2C clock source is the HSI16 oscillator.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wakeup from Stop mode feature.

Caution: If wakeup from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

32.4.16 Error conditions

The following errors are the error conditions which may cause communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte should be sent. The content of the I2C_TXDR register is sent if TXE=0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Packet error checking error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to *Section 32.3: I2C implementation*.

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Timeout Error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 32.3: I2C implementation*.

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{LOW:MEXT}$ parameter)
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus $t_{LOW:SEXT}$ parameter)

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 32.3: I2C implementation*.

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

32.4.17 DMA requests

Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see *Section 11: Direct memory access controller (DMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Master transmitter*.
- In slave mode:
  - With NOSTRETCH = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH = 1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to *SMBus slave transmitter* and *SMBus master transmitter*.

Note: If DMA is used for transmission, the TXIE bit does not need to be enabled.

Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In Master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the

DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

- In Slave mode with NOSTRETCH = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see *Section 32.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to *SMBus Slave receiver* and *SMBus master receiver*.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

32.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_ configuration bits in the DBG module.

32.5 I2C low-power modes

Table 224. Effect of low-power modes on the I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The I2C registers content is kept(2). If WUPEN = 1 and I2C is clocked by an internal oscillator (HSI16): the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. If WUPEN = 0: the I2C must be disabled before entering Stop mode
Standby	The I2C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 32.3* for information about the Stop modes supported by each instance. If wakeup from a specific Stop mode is not supported, the instance must be disabled before entering this Stop mode.
2. In Stop 2 mode, only I2C3 register content is kept. The I2C1 and I2C2 instances are powered down and must be reinitialized after exiting Stop 2 mode.

32.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 225. I2C Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop 0, Stop 1, Stop 2 mode	Exit the Standby, Shutdown mode
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	No
		Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
		Stop detection interrupt flag	STOPF	STOPIE	Write STOPCF=1			
		Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0			
		Transfer complete	TC		Write START=1 or STOP=1			
		Address matched	ADDR	ADDRIE	Write ADDRCF=1		Yes(1)	
		NACK reception	NACKF	NACKIE	Write NACKCF=1		No	
	I2C_ER	Bus error	BERR	ERRIE	Write BERRCF=1	Yes	No	No
		Arbitration loss	ARLO		Write ARLOCF=1			
		Overrun/ Underrun	OVR		Write OVRCF=1			
		PEC error	PECERR		Write PECERRCF=1			
		Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF=1			
		SMBus alert	ALERT		Write ALERTCF=1			

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the Wakeup from Stop mode feature. Refer to *Section 32.3: I2C implementation*.

32.7 I2C registers

Refer to *Section 1.2 on page 55* for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

32.7.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **PECEN:** PEC enable
- 0: PEC calculation disabled
- 1: PEC calculation enabled

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 22 **ALERTEN**: SMBus alert enable
- 0: The SMBus alert pin (SMBA) is not supported in host mode (SMBHEN=1). In device mode (SMBHEN=0), the SMBA pin is released and the Alert Response Address header is disabled (0001100x followed by NACK).
- 1: The SMBus alert pin is supported in host mode (SMBHEN=1). In device mode (SMBHEN=0), the SMBA pin is driven low and the Alert Response Address header is enabled (0001100x followed by ACK).

Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 21 **SMBDEN**: SMBus device default address enable
- 0: Device default address disabled. Address 0b1100001x is NACKed.
- 1: Device default address enabled. Address 0b1100001x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 20 **SMBHEN**: SMBus host address enable
- 0: Host address disabled. Address 0b0001000x is NACKed.
- 1: Host address enabled. Address 0b0001000x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 19 **GCEN**: General call enable
0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wakeup from Stop mode enable
0: Wakeup from Stop mode disable.
1: Wakeup from Stop mode enable.
Note: If the Wakeup from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.
Note: WUPEN can be set only when DNF = '0000'

Bit 17 **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.
0: Clock stretching enabled
1: Clock stretching disabled
Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 16 **SBC**: Slave byte control
This bit is used to enable hardware byte control in slave mode.
0: Slave byte control disabled
1: Slave byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to 1 t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to15 t_{I2CCLK}
Note: If the analog filter is also enabled, the digital filter is added to the analog filter. This filter can only be programmed when the I2C is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generate an interrupt:
Arbitration Loss (ARLO)
Bus Error detection (BERR)
Overrun/Underrun (OVR)
Timeout detection (TIMEOUT)
PEC error detection (PECERR)
Alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer Complete interrupt enable
0: Transfer Complete interrupt disabled
1: Transfer Complete interrupt enabled
Note: Any of these events generate an interrupt:
Transfer Complete (TC)
Transfer Complete Reload (TCR)

Bit 5 **STOPIE**: Stop detection Interrupt enable
0: Stop detection (STOPF) interrupt disabled
1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (slave only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disable
1: Peripheral enable
Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

32.7.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_ WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.

0: No PEC transfer.

1: PEC transmission/reception is requested

Note: Writing '0' to this bit has no effect.

This bit has no effect when RELOAD is set.

This bit has no effect is slave mode when SBC=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'.

Refer to Section 32.3: I2C implementation.

Bit 25 **AUTOEND**: Automatic end mode (master mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in slave mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing '0' to this bit has no effect.

This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

In Master mode:

0: No Stop generation.

1: Stop generation after current byte transfer.

Note: Writing '0' to this bit has no effect.

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by an address matched in slave mode, by a timeout error detection, or when PE = 0.

0: No Start generation.

1: Restart/Start generation:

If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated start condition when RELOAD=0, after the end of the NBYTES transfer.

Otherwise setting this bit generates a START condition once the bus is free.

Note: Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.

1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (master mode)

0: The master operates in 7-bit addressing mode,

1: The master operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (master mode)

0: Master requests a write transfer.

1: Master requests a read transfer.

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Slave address (master mode)

In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] should be written with the 7-bit slave address to be sent. The bits SADD[9], SADD[8] and SADD[0] are don't care.

In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] should be written with the 10-bit slave address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

32.7.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received slave address OA1 is NACKed.
1: Own address 1 enabled. The received slave address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN=0.

Bits 9:0 **OA1[9:0]**: Interface own slave address

7-bit addressing mode: OA1[7:1] contains the 7-bit own slave address. The bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.

Note: These bits can be written only when OA1EN=0.

32.7.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable
0: Own address 2 disabled. The received slave address OA2 is NACKed.
1: Own address 2 enabled. The received slave address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN=0.

As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.

32.7.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to *I2C timings on page 953*) and for SCL high and low level counters (refer to *I2C master initialization on page 968*).

t_{PRESC} = (PRESC+1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

t_{SCLH} = (SCLH+1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

t_{SCLL} = (SCLL+1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

32.7.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable
0: Extended clock timeout detection is disabled
1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B
This field is used to configure the cumulative clock extension timeout:
In master mode, the master cumulative clock low extend time ($t_{LOW:MEXT}$) is detected
In slave mode, the slave cumulative clock low extend time ($t_{LOW:SEXT}$) is detected
$t_{LOW:EXT}$= (TIMEOUTB+1) x 2048 x t_{I2CCLK}
Note: These bits can be written only when TEXTEN=0.

Bit 15 **TIMOUTEN**: Clock timeout enable
0: SCL timeout detection is disabled
1: SCL timeout detection is enabled: when SCL is low for more than $t_{TIMEOUT}$ (TIDLE=0) or high for more than t_{IDLE} (TIDLE=1), a timeout error is detected (TIMEOUT=1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection
0: TIMEOUTA is used to detect SCL low timeout
1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)
Note: This bit can be written only when TIMOUTEN=0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A
This field is used to configure:
The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE=0
$t_{TIMEOUT}$= (TIMEOUTA+1) x 2048 x t_{I2CCLK}
The bus idle condition (both SCL and SDA high) when TIDLE=1
t_{IDLE}= (TIMEOUTA+1) x 4 x t_{I2CCLK}
Note: These bits can be written only when TIMOUTEN=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 32.3: I2C implementation.

32.7.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (Slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 **DIR**: Transfer direction (Slave mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 11 **PECERR**: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 10 **OVR**: Overrun/Underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting *BERRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for master mode, or for slave mode when the SBC bit is set.

Bit 6 **TC**: Transfer Complete (master mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: Stop detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software when NOSTRETCH = 1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

32.7.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 11 **PECCF**: PEC Error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.

Bit 10 **OVRCF**: Overrun/Underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

32.7.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]:** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 32.3: I2C implementation.

32.7.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data
Data byte received from the I^2C bus

32.7.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data
Data byte to be transmitted to the I^2C bus
Note: These bits can be written only when TXE = 1.

32.7.12 I2C register map

The table below provides the I2C register map and reset values.

Table 226. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	I2C_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x4	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0xC	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_ TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_ TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 226. I2C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.4 on page 62* for the register boundary addresses.

33 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

33.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and Half-duplex Single-wire communications, as well as LIN (local interconnection network), Smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and Modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

33.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection
- Wakeup from Stop mode

33.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for Smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

33.4 USART implementation

The table below describes USART implementation on STM32WLEx devices. It also includes LPUART for comparison.

Table 227. USART / LPUART features

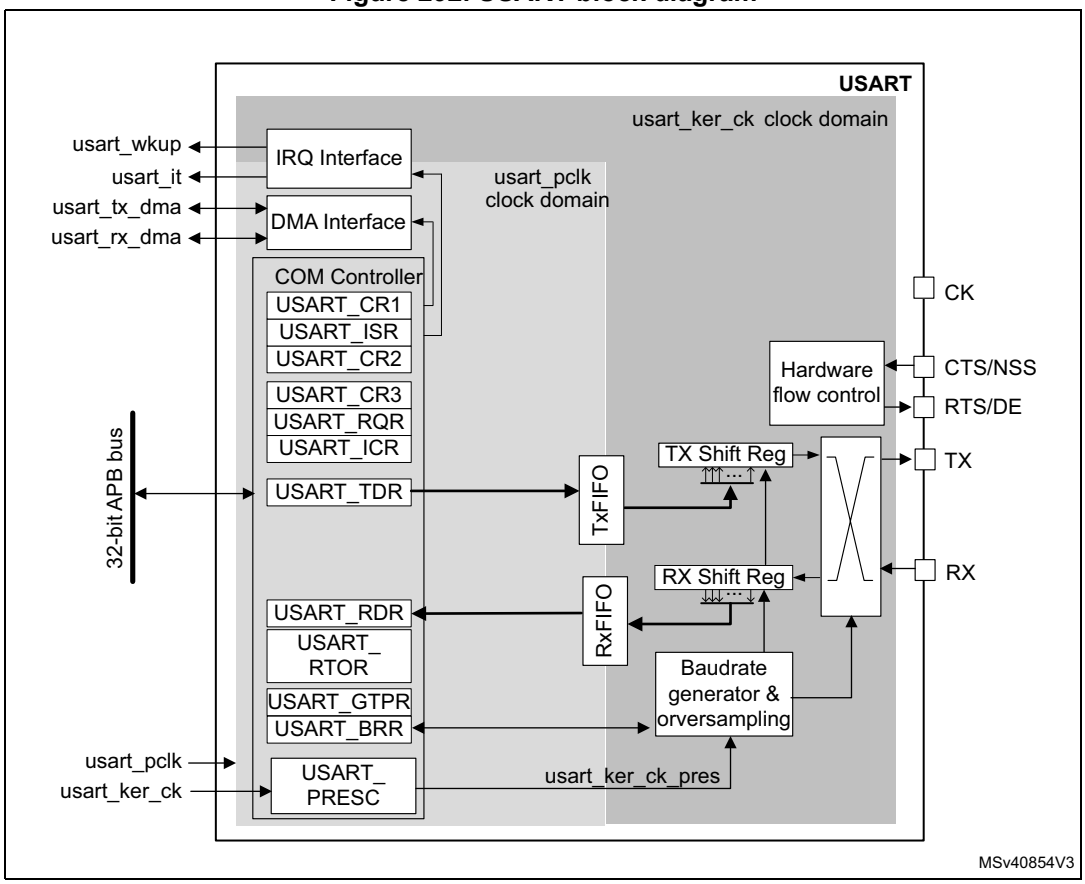
USART / LPUART modes/features(1)	USART1/2	LPUART1
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode (Master/Slave)	X	-
Smartcard mode	X	-
Single-wire Half-duplex communication	X	X
IrDA SIR ENDEC block	X	-
LIN mode	X	-
Dual clock domain and wakeup from low-power mode	X	X
Receiver timeout interrupt	X	-
Modbus communication	X	-
Auto baud rate detection	X	-
Driver Enable	X	X
USART data length	7, 8 and 9 bits	
Tx/Rx FIFO	X	X
Tx/Rx FIFO size	8	
Wakeup from Stop mode	X(2)	X(3)

1. X = supported.
2. Wakeup supported from Stop 0 and Stop 1 modes.
3. Wakeup supported from Stop 0, Stop 1 and Stop 2 modes.

33.5 USART functional description

33.5.1 USART block diagram

Figure 292. USART block diagram



The simplified block diagram given in *Figure 292* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.

- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is disabled, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

33.5.2 USART signals

USART bidirectional communications

USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)

  RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

- **TX** (Transmit Data Output)

  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In Single-wire and Smartcard modes, this I/O is used to transmit and receive data.

RS232 Hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS** (Clear To Send)

  When driven high, this signal blocks the data transmission at the end of the current transfer.

- **RTS** (Request To Send)

  When it is low, this signal indicates that the USART is ready to receive data.

RS485 Hardware control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)

  This signal activates the transmission mode of the external transceiver.

Note: DE and RTS share the same pin.

Synchronous master/slave mode and Smartcard mode

The following pin is required in synchronous master/slave mode and Smartcard mode:

- CK

  This pin acts as Clock output in Synchronous master and Smartcard modes.

  It acts as Clock input is Synchronous slave mode.

  In Synchronous Master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.

  In Smartcard mode, CK output provides the clock to the smartcard.

- NSS

  This pin acts as Slave Select input in Synchronous slave mode.

Note: NSS and CTS share the same pin.

33.5.3 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 293*):

- 7-bit character length: M[1:0] = ‘10’
- 8-bit character length: M[1:0] = ‘00’
- 9-bit character length: M[1:0] = ‘01’

Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

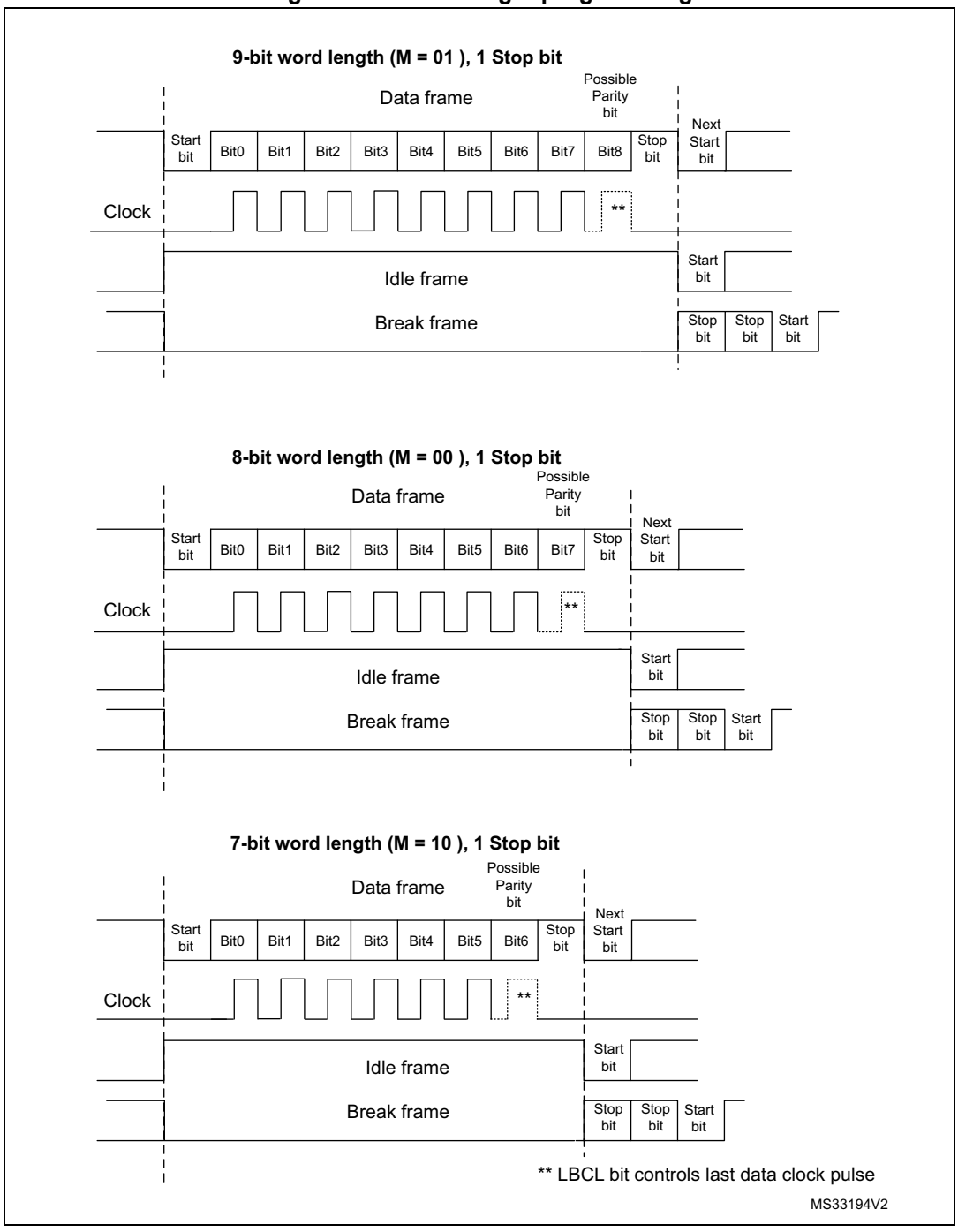
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 293. Word length programming



33.5.4 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and Smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART_CR3 control register.

In this case:

- The RXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

  This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

  RXFTCFG data have been received: one data in USART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101’, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data is not set the overrun flag.
- The TXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

33.5.5 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

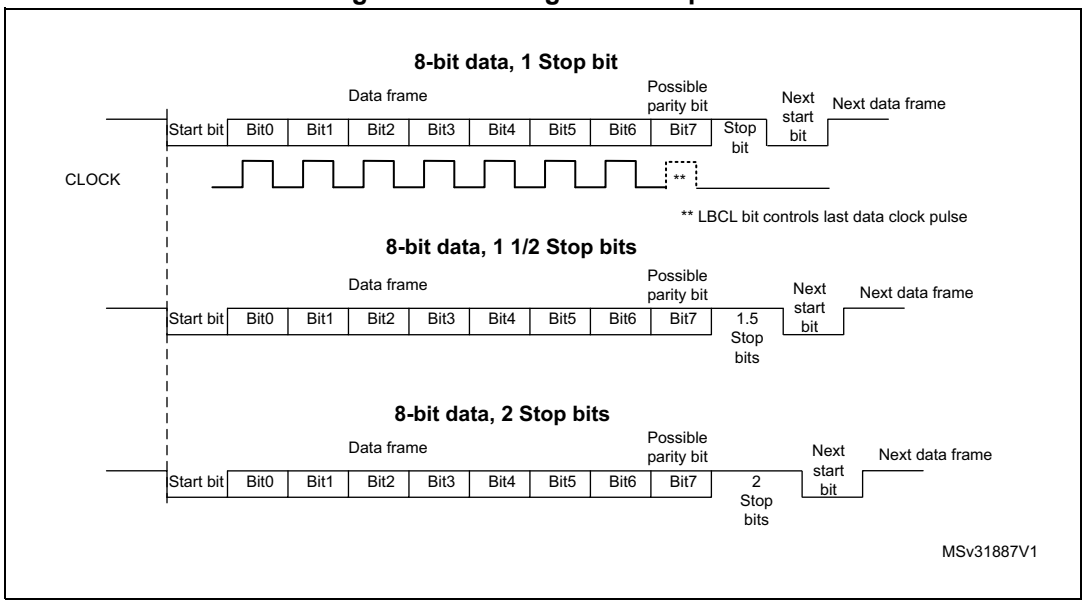
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, Single-wire and Modem modes.
- ***1.5 stop bits***: To be used in Smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission features 10 low bits (when M[1:0] = ‘00’) or 11 low bits (when M[1:0] = ‘01’) or 9 low bits (when M[1:0] = ‘10’) followed by 2 stop bits (see *Figure 294*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 294. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 33.5.10: USART multiprocessor communication*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:

  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;

  – the USART_TDR register is empty;

  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:

  – the TXFIFO is not full;

  – the USART_TDR register is empty;

  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

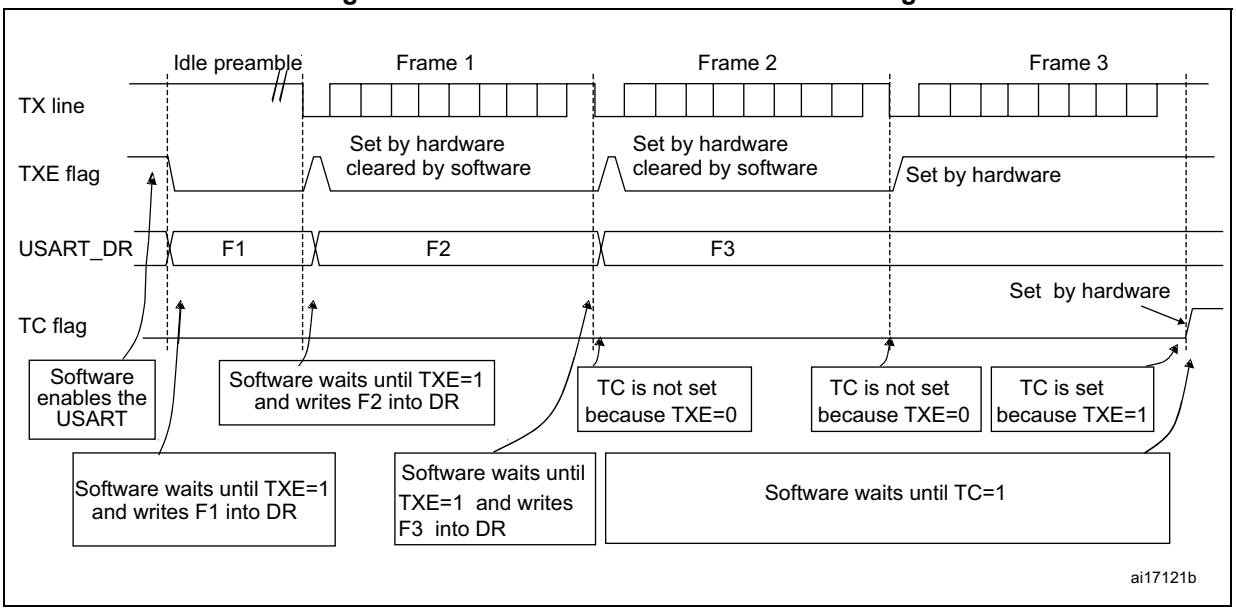
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the device to enter the low-power mode (see *Figure 295: TC/TXE behavior when transmitting*).

Figure 295. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 293*).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

33.5.6 USART receiver

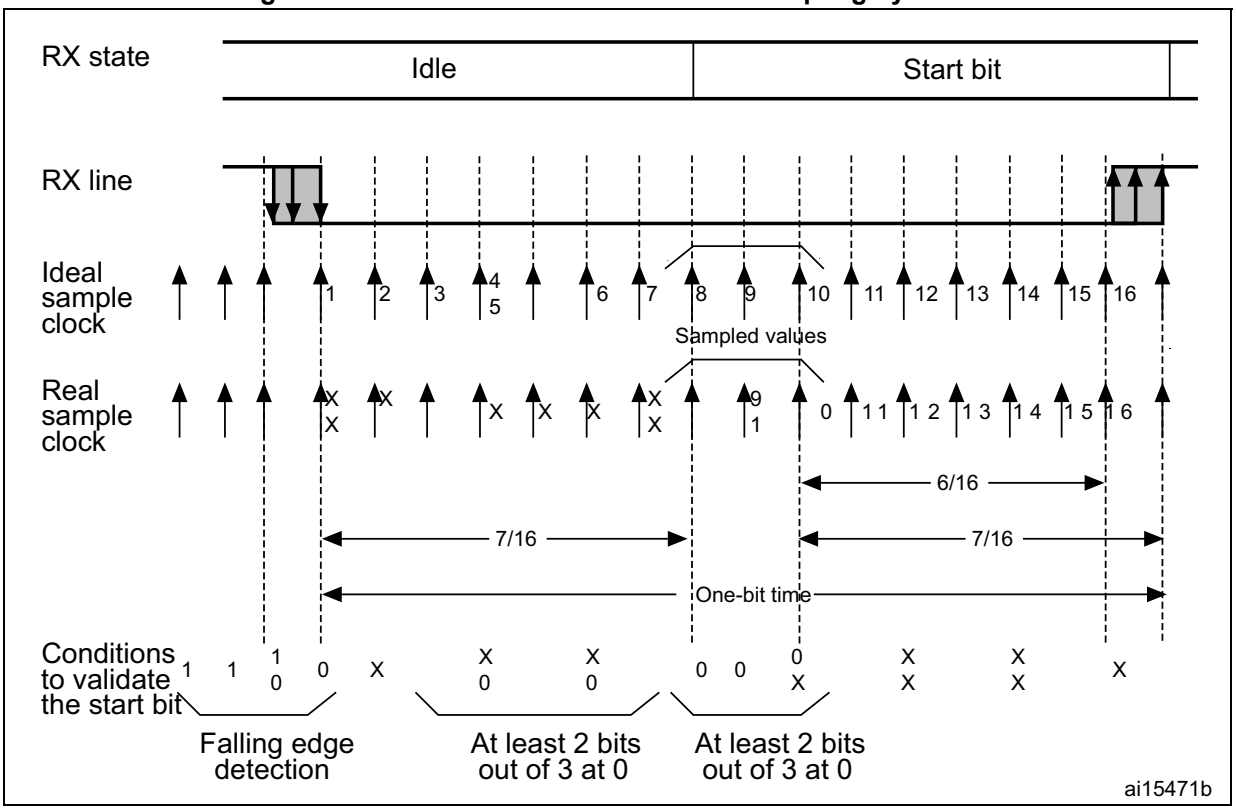
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 296. Start bit detection when oversampling by 16 or 8



Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at ‘0’ (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at ‘0’ and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at ‘0’).

The start bit is validated but the NE noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at ‘0’ (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at ‘0’.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 33.5.10: USART multiprocessor communication*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to ‘1’ in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to ‘1’ in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see Section *Reset and clock control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

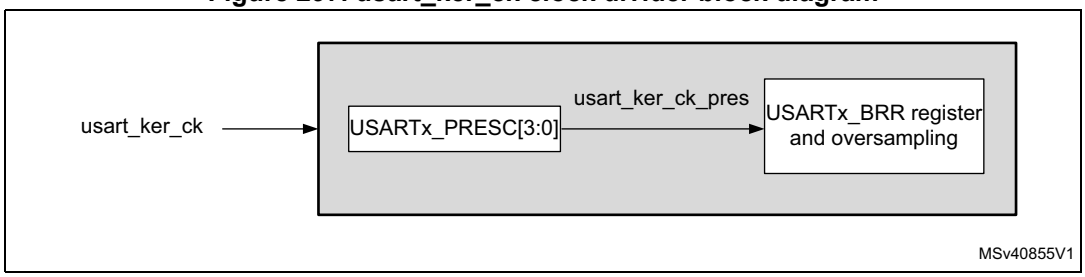
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wakeup from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see Section *Reset and clock control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 297. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 298* and *Figure 299*).

Depending on your application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1034*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum

usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on your application:

  – select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Figure 228*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1034*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NECF bit in USART_ICR register.

Note: Noise error is not supported in SPI mode.

Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to '0 ' by hardware.

Figure 298. Data sampling when oversampling by 16

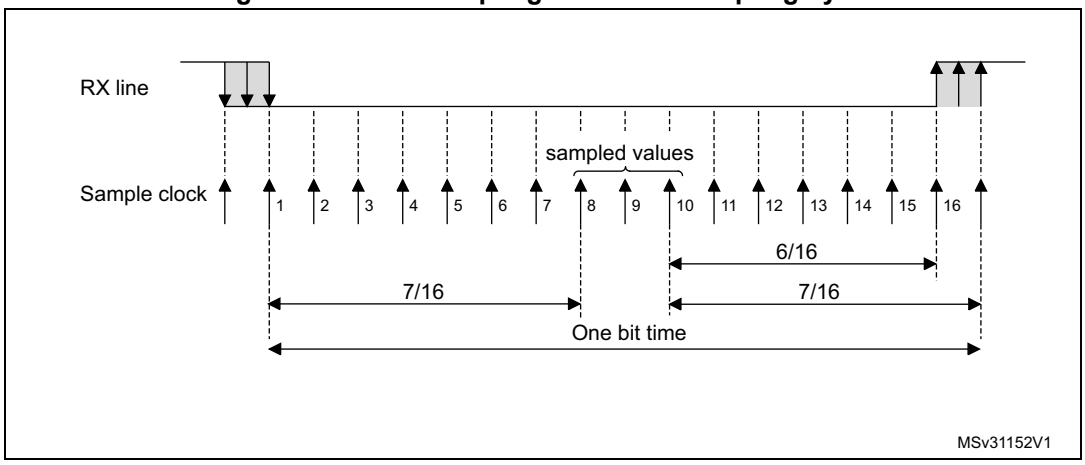


Figure 299. Data sampling when oversampling by 8

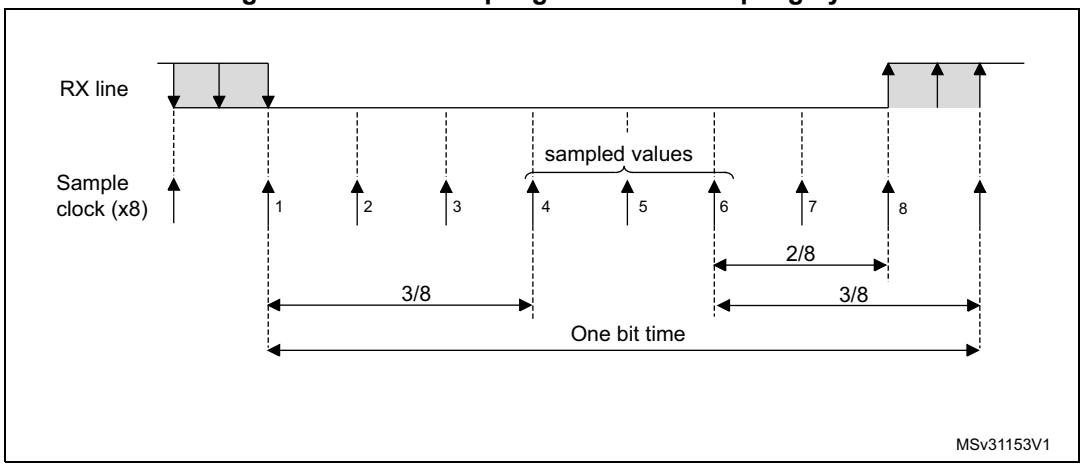


Table 228. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing ‘1’ to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

- **0.5 stop bit (reception in Smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- 1.5 stop bits (Smartcard mode)

  When transmitting in Smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the Smartcard has detected a parity error.

  In the event of a parity error, the Smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 33.5.16: USART receiver timeout on page 1048* for more details).
- 2 stop bits

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

33.5.7 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = ‘0’ or ‘1’)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ckpres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart\_ker\_ckpres}}{\text{USARTDIV}}$$

Equation 2: baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ckpres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 baud with usart_ker_ck_pres = 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >> 1 = 0x1
  BRR = 0x681

Example 2

To obtain 921.6 Kbaud with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR = USARTDIV = 0d52 = 0x34
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR = 0x64

33.5.8 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 229*, *Table 230*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 229. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 230. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 229 and Table 230 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

33.5.9 USART Auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at ‘1’.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a ‘0’).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

33.5.10 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise Mute mode might remain active.

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to ‘1’. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

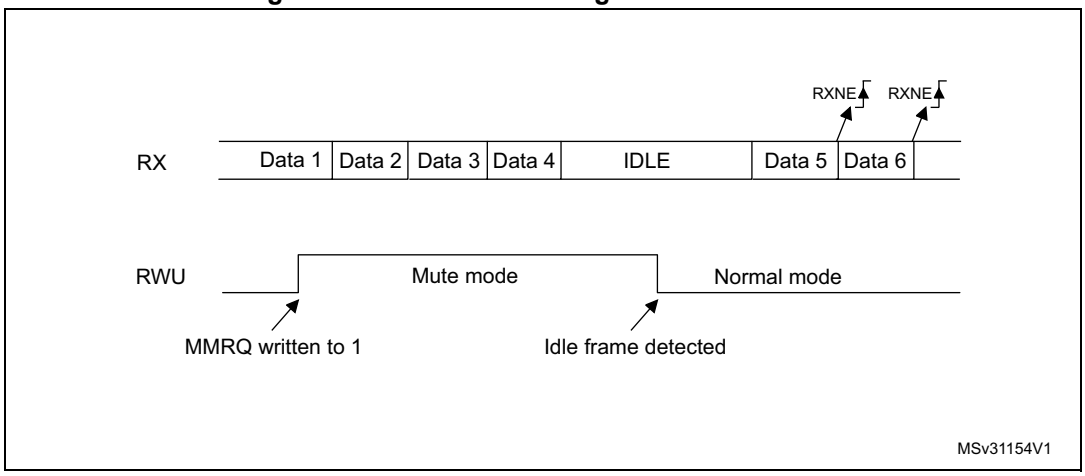
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters Mute mode when the MMRQ bit is written to ‘1’ and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of Mute mode behavior using Idle line detection is given in *Figure 300*.

Figure 300. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, Mute mode is not entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters Mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering Mute mode.

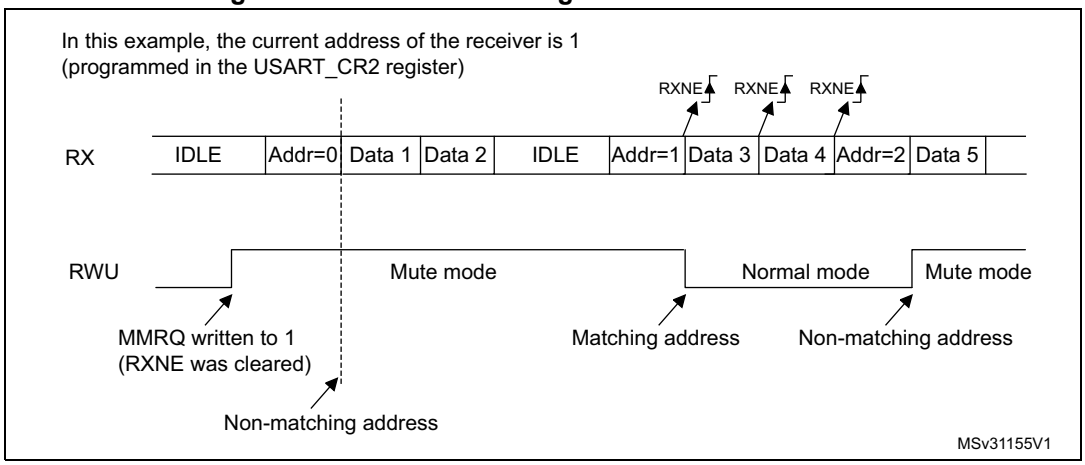
The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in Mute mode

An example of Mute mode behavior using address mark detection is given in *Figure 301*.

Figure 301. Mute mode using address mark detection



33.5.11 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a Half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

33.5.12 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 231*.

Table 231. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits are set, the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS=1).

33.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 33.4: USART implementation on page 1017*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USART_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 33.5.4* has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0 bits as a break character. Then two bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as ‘0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (i.e. stop bit detected at ‘0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the *Figure 302: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 1043*.

Examples of break frames are given on *Figure 303: Break detection in LIN mode vs. Framing error detection on page 1044*.

Figure 302. Break detection in LIN mode (11-bit break length - LBDL bit is set)

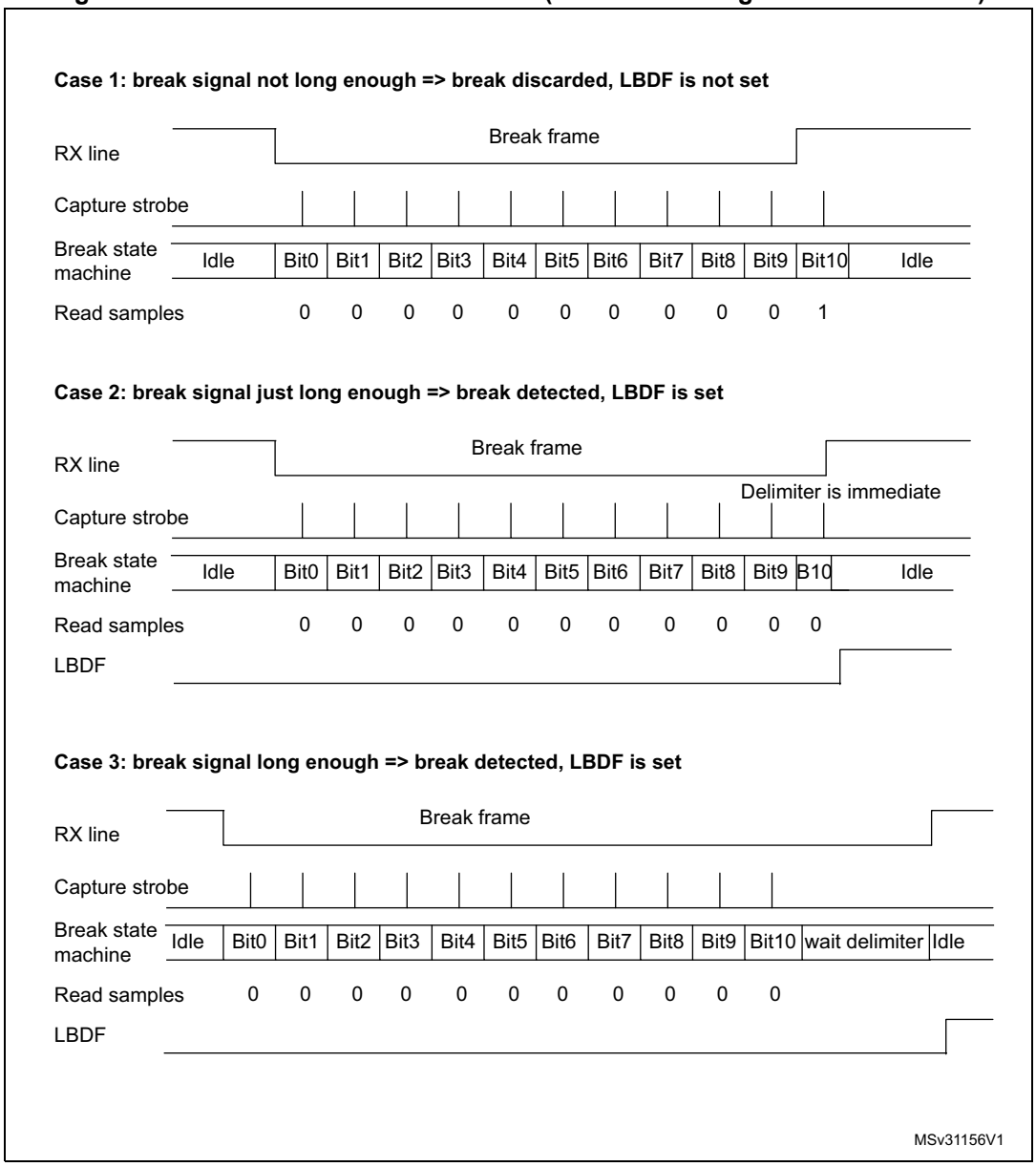
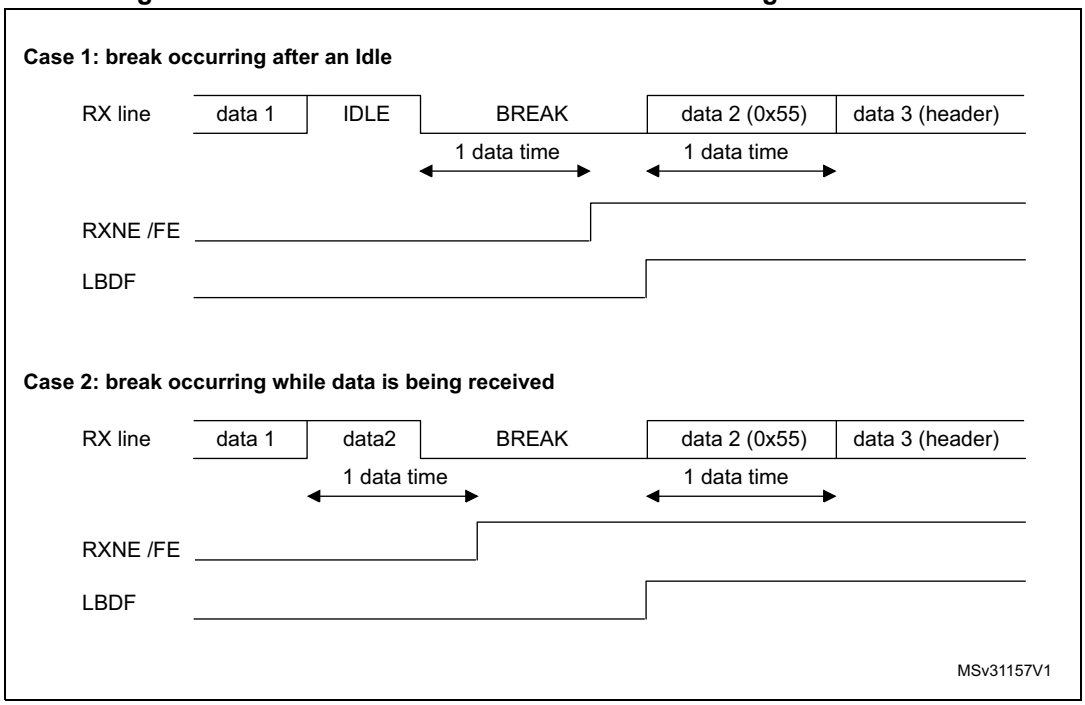


Figure 303. Break detection in LIN mode vs. Framing error detection



33.5.14 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to '1'. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 304*, *Figure 305* and *Figure 306*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 304. USART example of synchronous master transmission

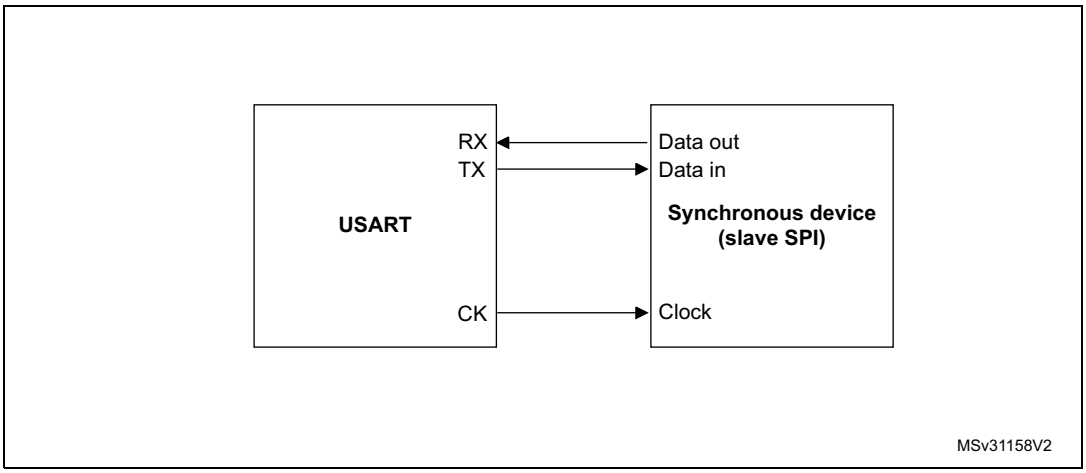


Figure 305. USART data clock timing diagram in synchronous master mode (M bits = 00)

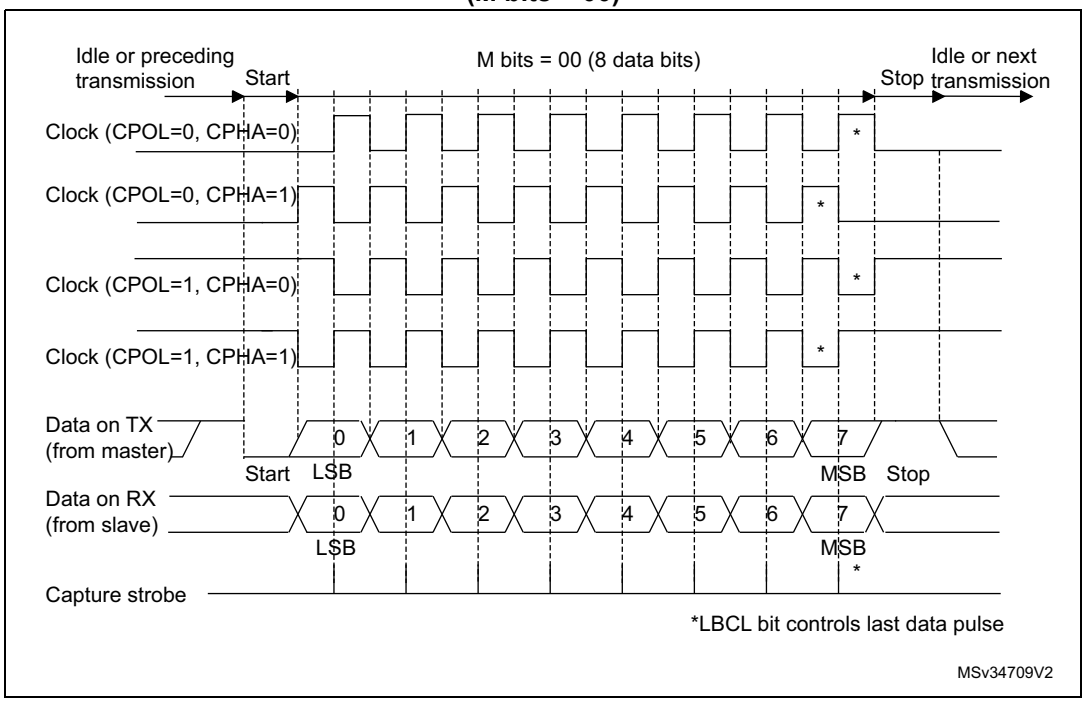
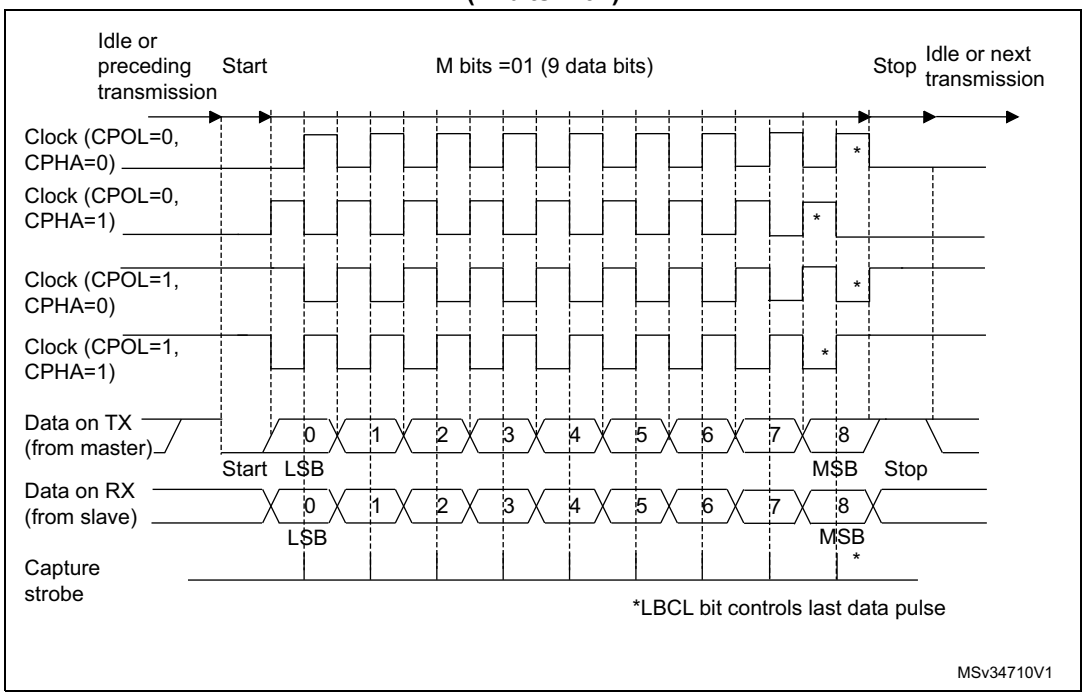


Figure 306. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to '1'. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

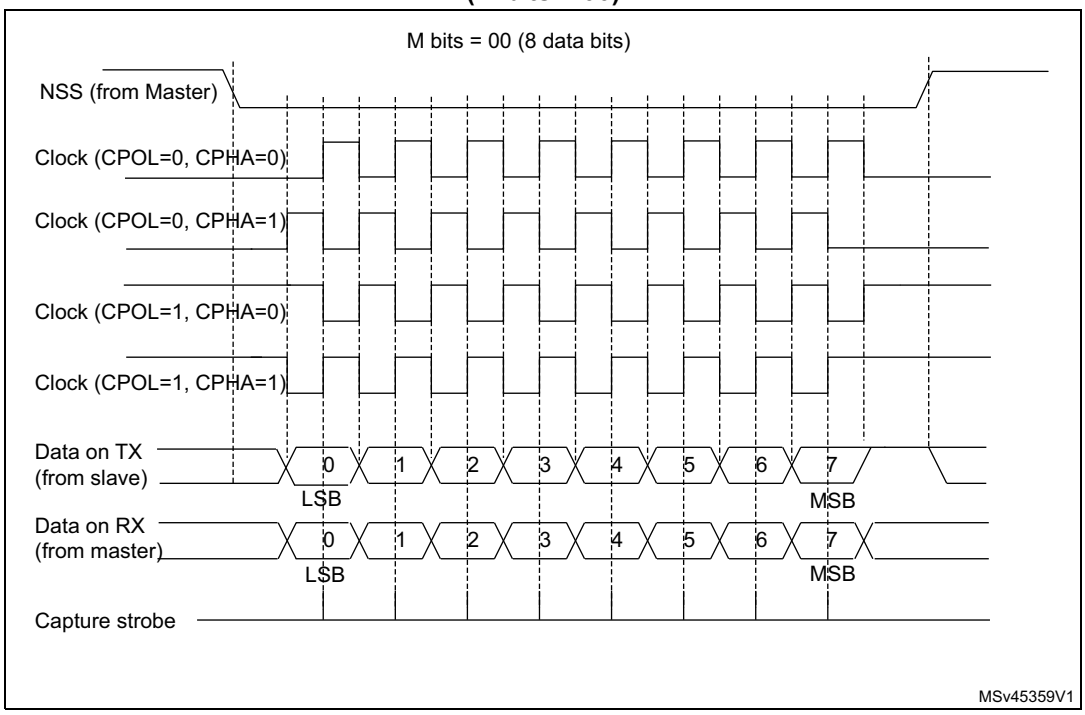
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 307*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 307. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  The SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI Slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR should be written 3 usart_ker_ck *cycles before the first CK edge.*

33.5.15 USART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

33.5.16 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = ‘00’ or STOP = ‘11’
- from the end of the second stop bit if STOP = ‘10’.
- from the beginning of the stop bit if STOP = ‘01’.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

33.5.17 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to *Section 33.4: USART implementation on page 1017*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the Smartcard.

The Smartcard interface is designed to support asynchronous Smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

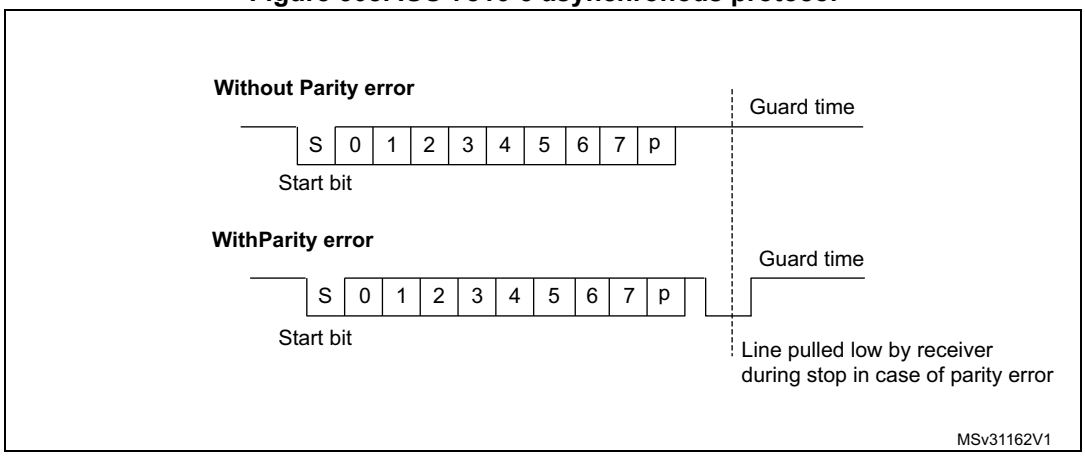
The USART should be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = ’11’ in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 308 shows examples of what can be seen on the data line with and without parity error.

Figure 308. ISO 7816-3 asynchronous protocol



When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

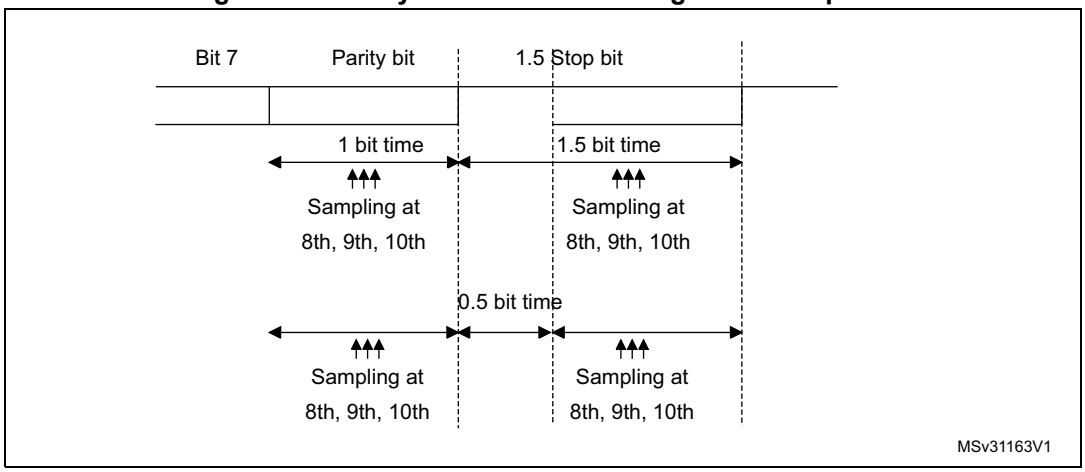
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The deassertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 309 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 309. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the Smartcard through the CK output. In Smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time -11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the Smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the Smartcard protocol definition, the BWT/CWT values should be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN = LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN = LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV = 1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, assuming that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH results in a USART received character of 03 and an odd parity.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART.

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

33.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 33.4: USART implementation on page 1017*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 310*).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbaud for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not

encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A ‘0‘ is transmitted as a high pulse and a ‘1’ is transmitted as a ‘0’. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 311*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to ‘1 stop bit’.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART_GTPR).

Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).

Figure 310. IrDA SIR ENDEC block diagram

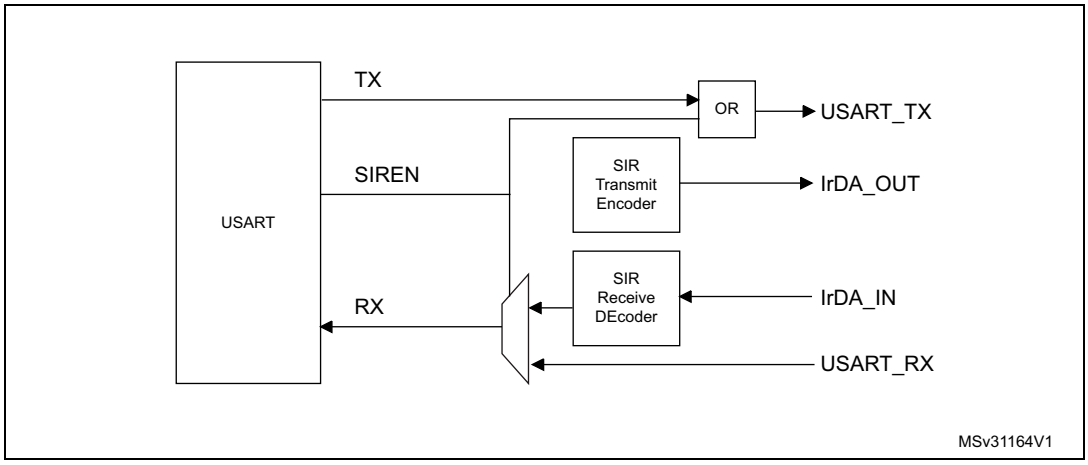
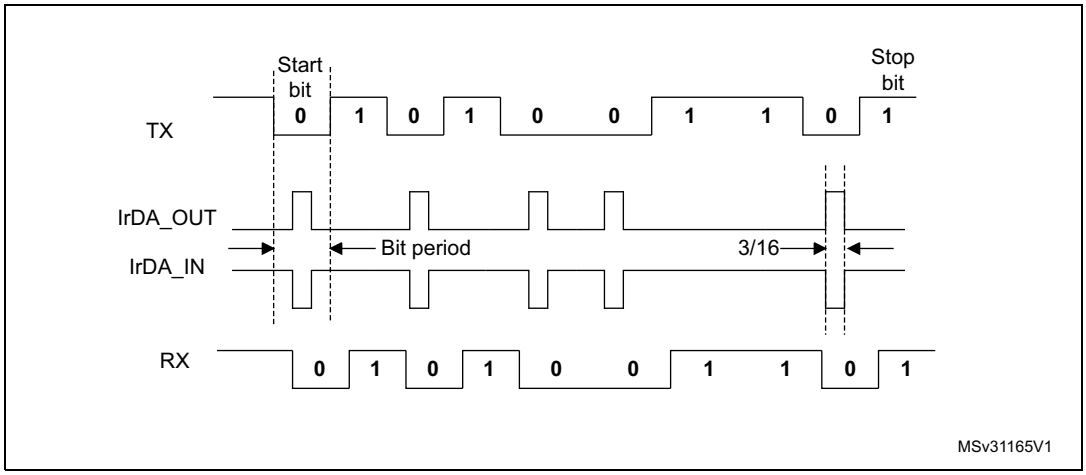


Figure 311. IrDA data modulation (3/16) - Normal mode



33.5.19 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 33.4: USART implementation on page 1017 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 33.5.6. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

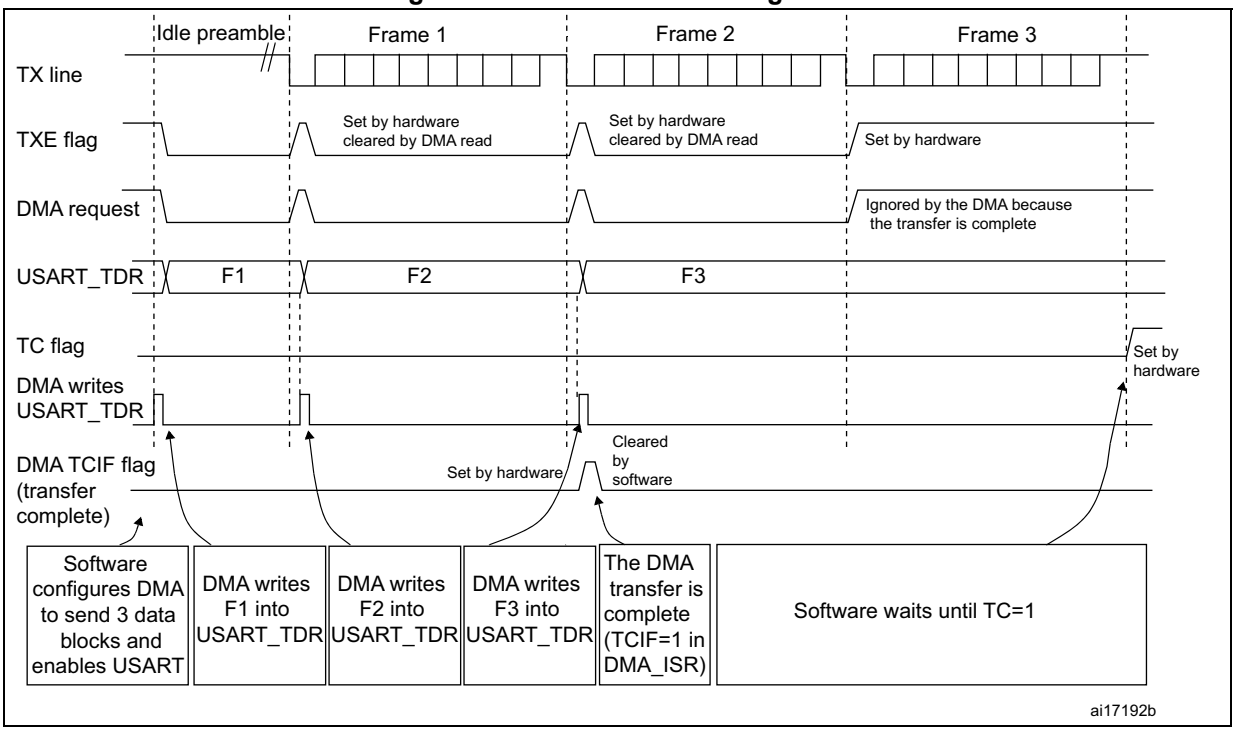
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 312. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

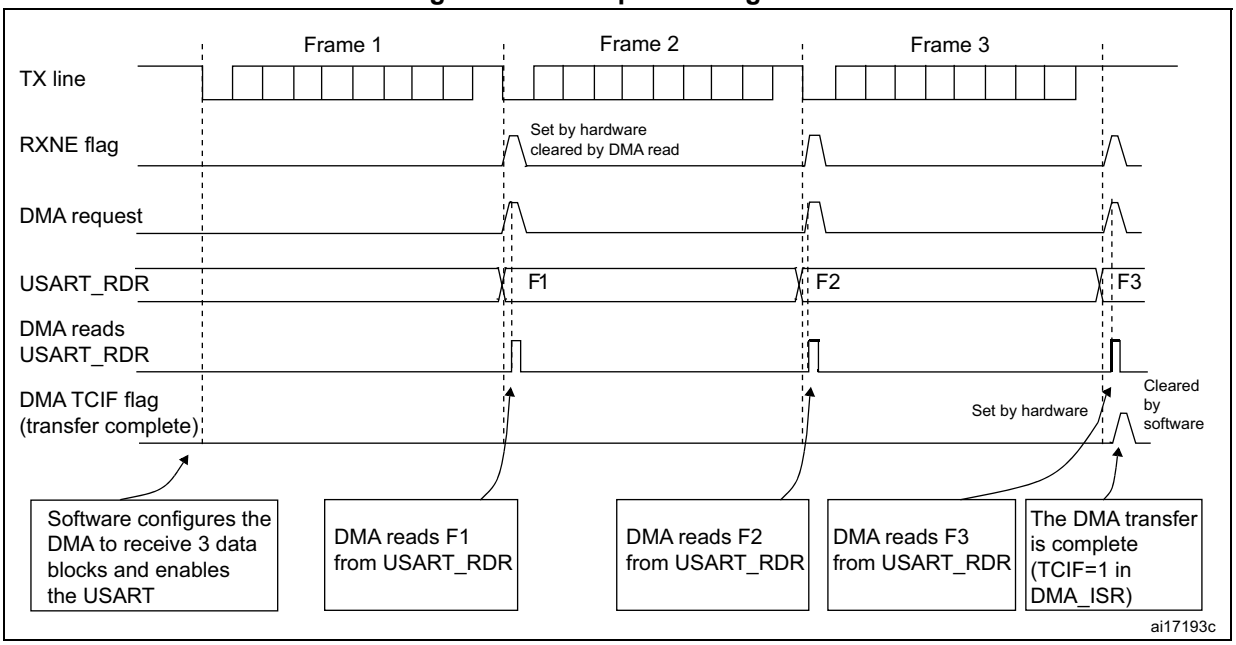
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 313. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

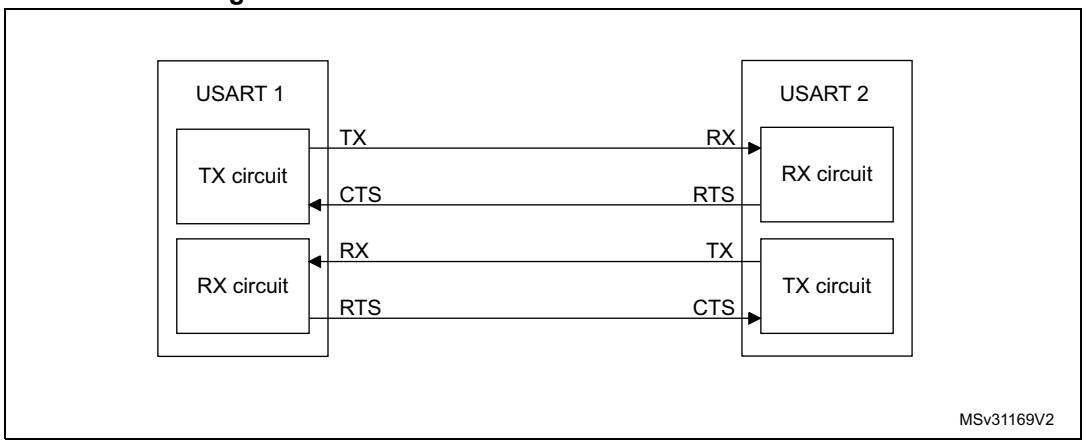
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

33.5.20 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The *Figure 314* shows how to connect 2 devices in this mode:

Figure 314. Hardware flow control between 2 USARTs

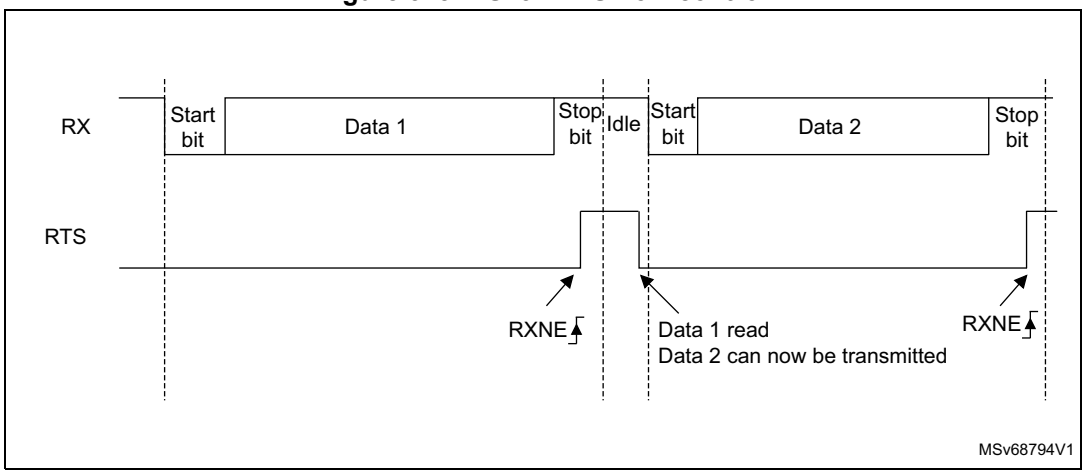


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to ‘1’ in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 315* shows an example of communication with RTS flow control enabled.

Figure 315. RS232 RTS flow control



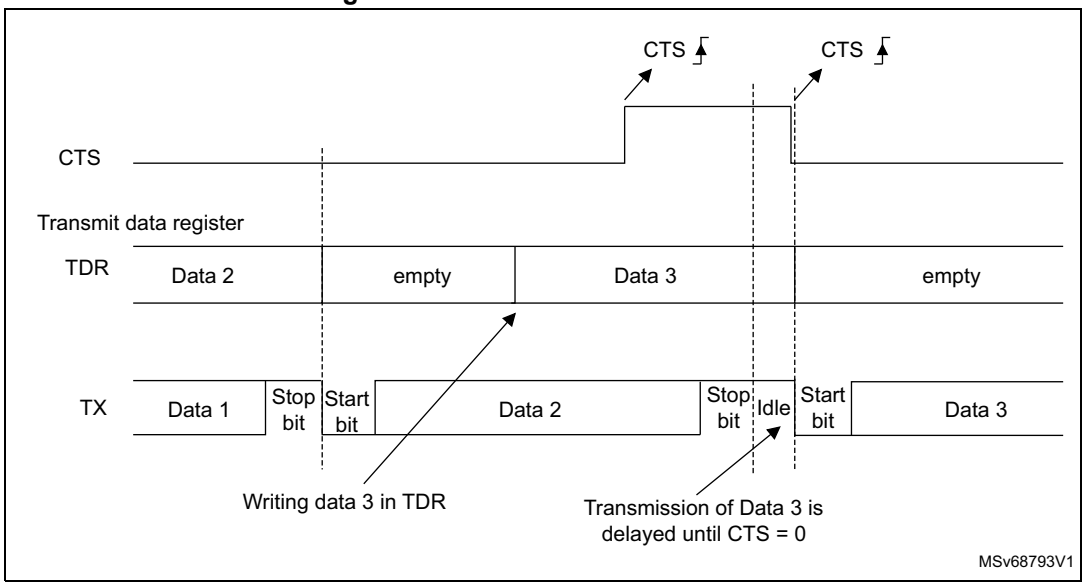
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 316* shows an example of communication with CTS flow control enabled.

Figure 316. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

33.5.21 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wakeup interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to ‘1’. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - to fill the TXFIFO
  - or to empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wakeup event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when Mute mode is enabled.

Using Mute mode with low-power mode

If the USART is put into Mute mode before entering low-power mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, Mute mode can be used with wakeup from low-power mode without any constraints (i.e.the two points mentioned above about Mute and low-power mode are valid only when FIFO management is disabled).

Wakeup from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wakeup event programmed to “address match detection” and FIFO management disabled.

Figure 317 shows the USART behavior when the wakeup event is verified.

Figure 317. Wakeup event verified (wakeup event = address match, FIFO disabled)

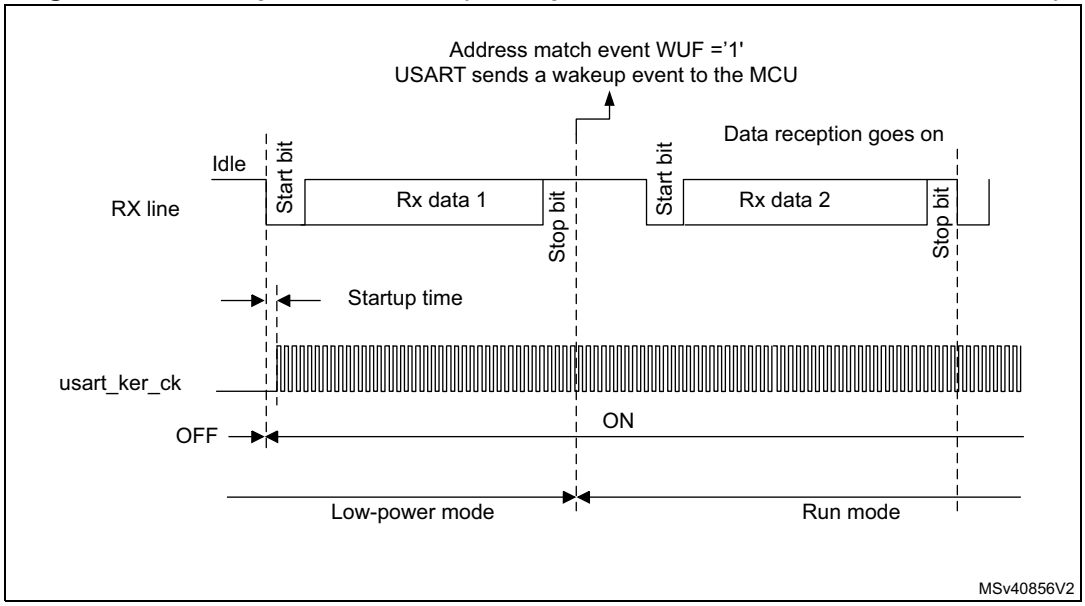
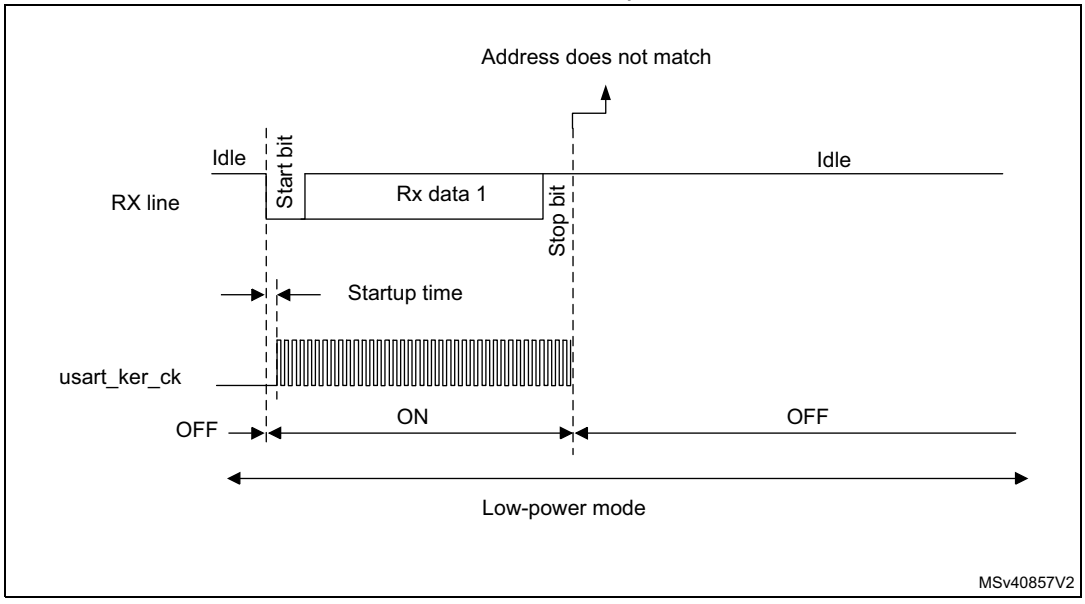


Figure 318 shows the USART behavior when the wakeup event is not verified.

Figure 318. Wakeup event not verified (wakeup event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as wakeup event. If the wakeup event is the start bit detection, the USART sends the wakeup event to the MCU at the end of the start bit.

Determining the maximum USART baud rate that enables to correctly wake up the device from low-power mode

The maximum baud rate that enables to correctly wake up the device from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 33.5.8: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 229: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the usart_ker_ck inaccuracy.

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% - 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wakeup correctly from low-power mode is: 1/11.32 µs = 88.36 Kbaud.

33.6 USART in low-power modes

Table 232. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept(2). The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 33.4: USART implementation* to know if the wakeup from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.
2. The content of the USART registers is kept only in Stop 0 and Stop 1 modes. In Stop 2 mode, the content of the USART registers is lost and must the reinitialized after exiting from Stop 2 mode.

33.7 USART interrupts

Refer to *Table 233* for a detailed description of all USART interrupt requests.

Table 233. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Transmission Complete Before Guard Time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RXNEIE/ RXFNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication	ORE(3)		Write 1 in ORECF		No	
	Framing Error in multibuffer communication	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wakeup from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the Wakeup from Stop mode feature. Refer to *Section 33.4: USART implementation* for the list of supported Stop modes.

2. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

33.8 USART registers

Refer to *Section 1.2 on page 55* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

33.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = ‘00’: 1 start bit, 8 Data bits, n Stop bit

M[1:0] = ‘01’: 1 start bit, 9 Data bits, n Stop bit

M[1:0] = ‘10’: 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE = 0).

Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of Block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 33.4: USART implementation on page 1017.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXFNF =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In Smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

33.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = ‘00’: 1 start bit, 8 Data bits, n Stop bit

M[1:0] = ‘01’: 1 start bit, 9 Data bits, n Stop bit

M[1:0] = ‘10’: 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE = 0).

Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of Block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 33.4: USART implementation on page 1017.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXE =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit (‘0’ followed by ‘1’) sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to ‘1’. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.
In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In Smartcard mode, (SCEN = 1), the CK pin is always available when CLKEN = 1, regardless of the UE bit value.

33.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in Mute mode or a character code to be recognized in low-power or Run mode:

- In Mute mode: they are used in multiprocessor communication to wakeup from Mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wakeup from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with Mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 and Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H). The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd = 0/mark)

1: TX pin signal values are inverted (V_{DD} =0/mark, Gnd = 1/idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd = 0/mark)

1: RX pin signal values are inverted (V_{DD} =0/mark, Gnd = 1/idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bits 13:12 **STOP[1:0]**: stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

In Smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 298* and *Figure 299*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 4 **ADDM7**: 7-bit Address Detection/4-bit Address Detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:

When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous Slave mode enable

When the SLVEN bit is set, the synchronous slave mode is enabled.

0: Slave mode disabled.

1: Slave mode enabled.

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

33.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS[1:0]		SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Remaining combinations: Reserved

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Remaining combinations: Reserved

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register
Note: If the USART does not support the Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF = 1 in the USART_ISR register
Note: WUFIE must be set before entering in low-power mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bits 21:20 **WUS[1:0]**: Wakeup from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in Smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmit mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 33.4: USART implementation on page 1017.

Bit 13 **DDRE**: DMA Disable on Reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred (used for Smartcard mode).

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data is written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables checking the communication flow w/o reading the data

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

33.8.5 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]**: USART baud rate

BRR[15:4]
BRR[15:4] = USARTDIV[15:4]
BRR[3:0]
When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].
When OVER8 = 1:
BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
BRR[3] must be kept cleared.

33.8.6 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bits 7:0 **PSC[7:0]**: Prescaler value

In IrDA low-power and normal IrDA mode:

PSC[7:0] = IrDA Normal and Low-Power baud rate

PSC[7:0] is used to program the prescaler for dividing the USART source clock to achieve the low-power frequency: the source clock is divided by the value given in the register (8 significant bits):

In Smartcard mode:

PSC[4:0] = Prescaler value

PSC[4:0] is used to program the prescaler for dividing the USART source clock to provide the Smartcard clock. The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value
00001: Divides the source clock by 1 (IrDA mode) / by 2 (Smarcard mode)
00010: Divides the source clock by 2 (IrDA mode) / by 4 (Smartcard mode)
00011: Divides the source clock by 3 (IrDA mode) / by 6 (Smartcard mode)
...
11111: Divides the source clock by 31 (IrDA mode) / by 62 (Smartcard mode)
0010 0000: Divides the source clock by 32 (IrDA mode)
...
1111 1111: Divides the source clock by 255 (IrDA mode)

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if Smartcard mode is used.

This bitfield is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Refer to Section 33.4: USART implementation on page 1017.

33.8.7 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the Block length in Smartcard T = 1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0: 0 information characters + LEC

BLEN = 1: 0 information characters + CRC

BLEN = 255: 254 information characters + CRC (total 256 characters))

In Smartcard mode, the Block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the Block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bits during which there is no activity on the RX line.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the Receiver timeout feature is not supported. Refer to Section 33.4: USART implementation on page 1017.

33.8.8 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing '1' to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in Smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO i.e. clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

33.8.9 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0X80 00C0

X = 2 if FIFO/Smartcard mode is enabled

X = 0 if FIFO is enabled and Smartcard mode is disabled

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of USART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the USART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in RXFTCFG in USART_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the USART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG threshold is configured to '101', RXFT flag is set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USART_RDR. Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is '1'. Refer to Section 33.4: USART implementation on page 1017.

Bit 24 **RXFF**: RXFIFO full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.

0: RXFIFO not full.

1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO empty

This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.

0: TXFIFO not empty.

1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit =1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1034).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the USART_RDR.

33.8.10 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is '1'. Refer to Section 33.4: USART implementation on page 1017.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register not full

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 or EIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1034).

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

33.8.11 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag
Writing 1 to this bit clears the UDRF flag in the USART_ISR register.
Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017

Bit 12 **EOBCF**: End of block clear flag
Writing 1 to this bit clears the EOBF flag in the USART_ISR register.
Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1017.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

33.8.12 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 292*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

33.8.13 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 292*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

33.8.14 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaining combinations: Reserved

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.

33.8.15 USART register map

The table below gives the USART register map and reset values.

Table 234. USART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1 FIFO enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			TCBGTIE	TXFTIE	WUFIE	WUS [1:0]		SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					X	X	X	X	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 234. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

34 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

34.1 LPUART introduction

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the device is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports Half-duplex Single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

34.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud to 9600 baud using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection

34.3 LPUART implementation

Below the description of LPUART implementation in comparison with USART.

Table 235. USART / LPUART features

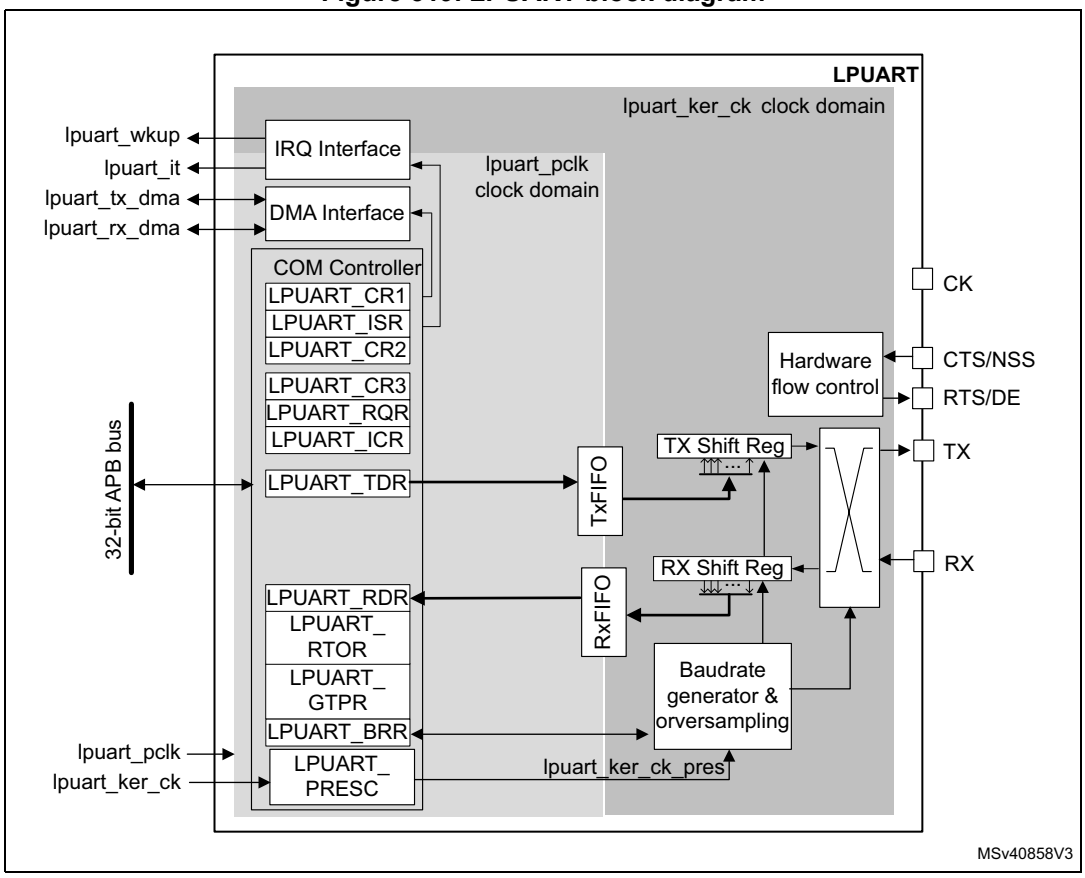
USART /LPUART modes/features(1)	USART1/2	LPUART1
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode (Master/Slave)	X	-
Smartcard mode	X	-
Single-wire Half-duplex communication	X	X
IrDA SIR ENDEC block	X	-
LIN mode	X	-
Dual clock domain and wakeup from low-power mode	X	X
Receiver timeout interrupt	X	-
Modbus communication	X	-
Auto baud rate detection	X	-
Driver Enable	X	X
USART data length	7, 8 and 9 bits	
Tx/Rx FIFO	X	X
Tx/Rx FIFO size	8	
Wakeup from Stop mode	X(2)	X(3)

1. X = supported.
2. Wakeup supported from Stop 0 and Stop 1 modes.
3. Wakeup supported from Stop 0, Stop 1 and Stop 2 modes.

34.4 LPUART functional description

34.4.1 LPUART block diagram

Figure 319. LPUART block diagram



The simplified block diagram given in *Figure 319* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.

- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is disabled, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

34.4.2 LPUART signals

LPUART bidirectional communications requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)

  RX is the serial data input.
- **TX (**Transmit Data Output)

  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In Single-wire mode, this I/O is used to transmit and receive the data.

RS232 hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS (**Clear To Send)

  When driven high, this signal blocks the data transmission at the end of the current transfer.
- **RTS** (Request to send)

  When it is low, this signal indicates that the USART is ready to receive data.

RS485 hardware flow control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)

  This signal activates the transmission mode of the external transceiver.

Note: DE and RTS share the same pin.

34.4.3 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 293*).

- 7-bit character length: M[1:0] = ‘10’
- 8-bit character length: M[1:0] = ‘00’
- 9-bit character length: M[1:0] = ‘01’

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

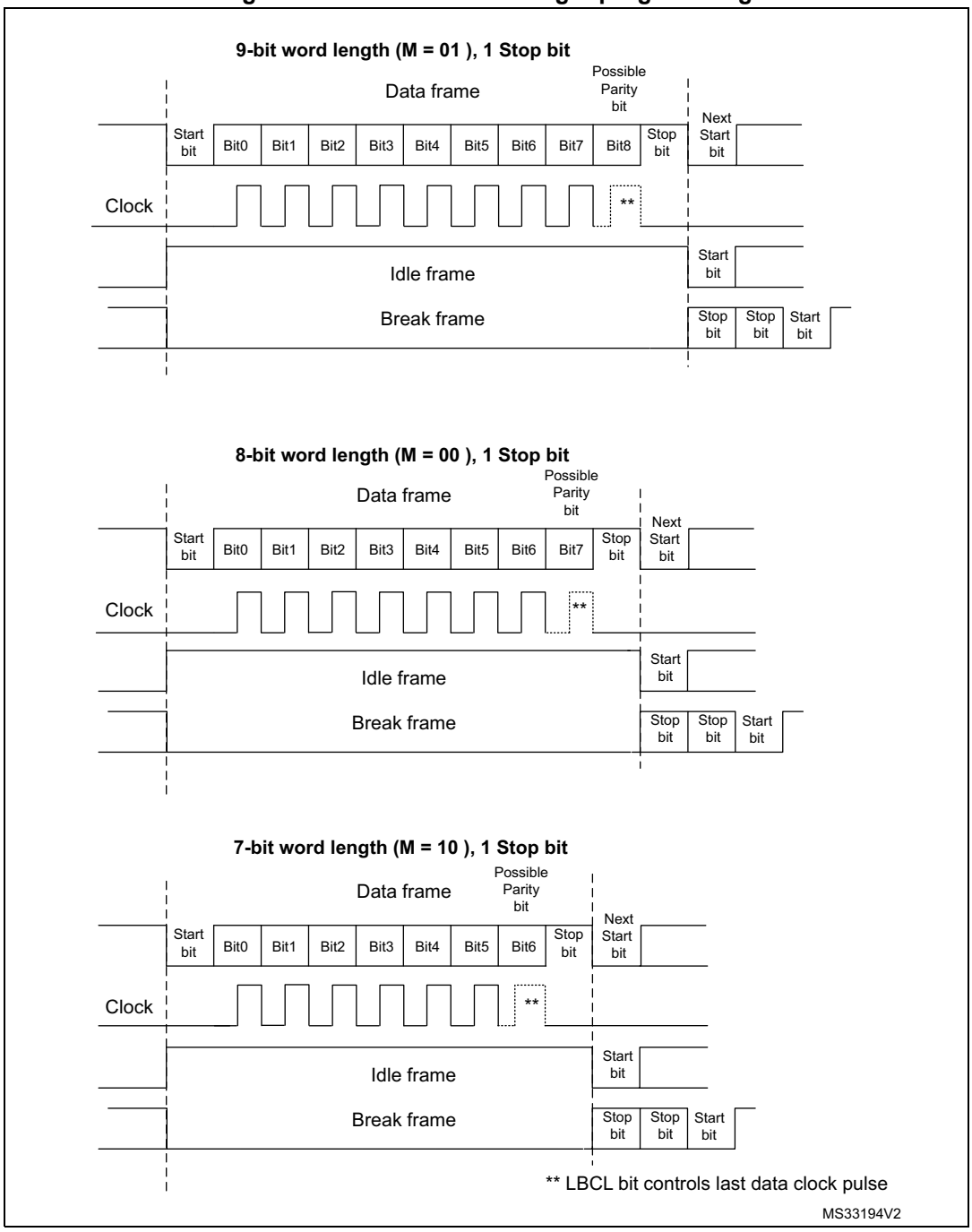
An ***Idle character*** is interpreted as an entire frame of “1”s. (The number of “1” ‘s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 320. LPUART word length programming



34.4.4 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in LPUART_CR1 register.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store

the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in LPUART_CR3 control register.

In this case:

- The RXFT flag is set in the LPUART_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

  This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

  RXFTCFG data have been received: one data in LPUART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to '101', the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size - 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.
- The TXFT flag is set in the LPUART_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

34.4.5 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 319*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

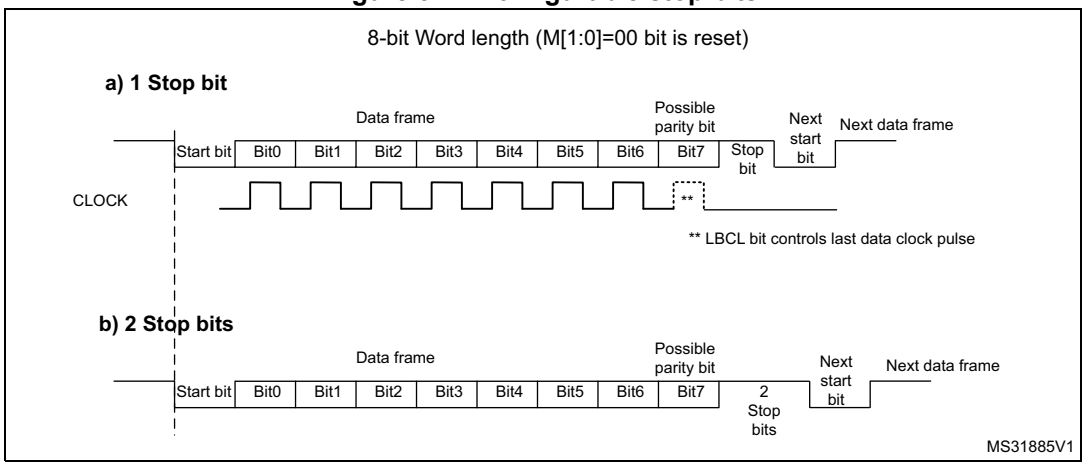
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13,12).

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 Stop bits***: This is supported by normal LPUART, Single-wire and Modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = ‘00’) or 11 low bits (when M[1:0] = ‘01’) or 9 low bits (when M[1:0] = ‘10’) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 321. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to ‘1’.
5. Select DMA enable (DMAT) in LPUART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in *Section 33.5.10: USART multiprocessor communication*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC = 1. This indicates that the transmission of the last frame is complete.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.

– When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:

  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;

  – the LPUART_TDR register is empty;

  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO Not Full) flag is set by hardware to indicate that:

  – the TXFIFO is not full;

  – the LPUART_TDR register is empty;

  – the next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

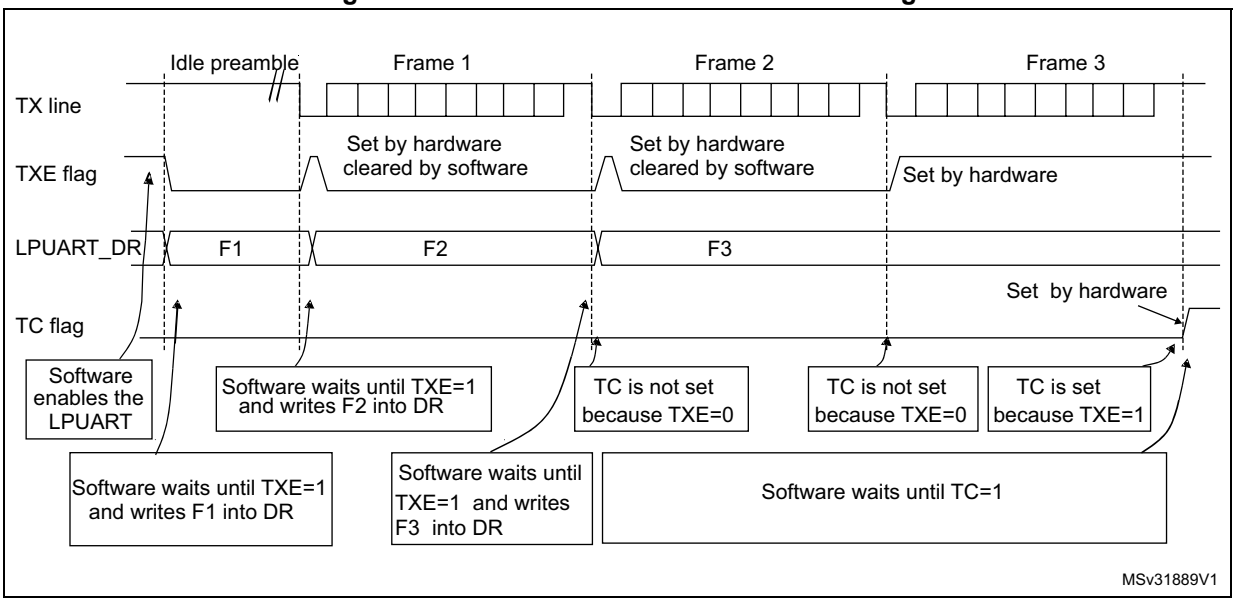
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

  After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the device to enter the low-power mode (see *Figure 322: TC/TXE behavior when transmitting*).

Figure 322. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 320*).

If a ‘1’ is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

34.4.6 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still ‘0’. If the start sample is at ‘1’, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 33.5.10: USART multiprocessor communication*.
6. Set the RE bit LPUART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is a data in the RXFIFO to be read.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.;
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data can not be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – the ORE bit is set;
  – the first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the Clock Control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

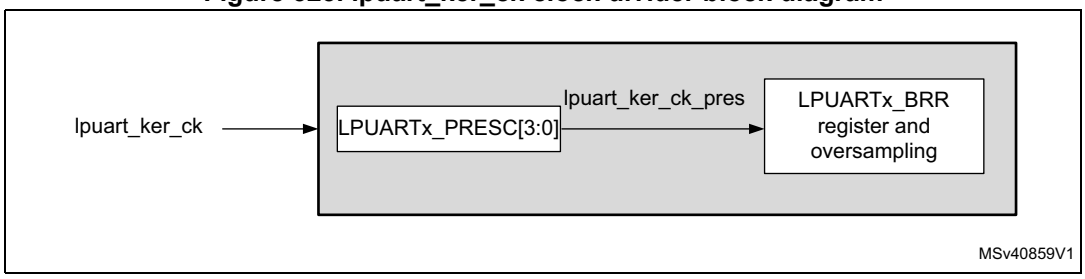
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wakeup from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 323. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the LPUART_RDR register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of

multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

34.4.7 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuartckpres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence the baud rate register value should not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

f_{CK} must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. For example, if the LPUART clock source frequency is 100 MHz, the maximum baud rate that can be reached is about 33 Mbaud.

Table 236. Error calculation for programmed baud rates at lpuart_ker_ck_pres = 32.768 kHz

Baud rate		lpuart_ker_ck_pres = 32.768 kHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	300 bauds	300 baud	0x6D3A	0
2	600 baud	600 baud	0x369D	0
3	1200 baud	1200.087 baud	0x1B4E	0.007
4	2400 baud	2400.17 baud	0xDA7	0.007
5	4800 baud	4801.72 baud	0x6D3	0.035
6	9600 baud	9608.94 baud	0x369	0.093

Table 237. Error calculation for programmed baud rates at f_{CK} = 100 MHz

Baud rate		f_{CK} = 100MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	38400 Baud	38400,04 Baud	A2C2A	0,0001
2	57600 Baud	57600,06 Baud	6C81C	0,0001
3	115200 Baud	115200,12 Baud	3640E	0,0001
4	230400 Baud	230400,23 Baud	1B207	0,0001
5	460800 Baud	460804,61 Baud	D903	0,001
6	921600 Baud	921625,81 Baud	6C81	0,0028
7	4000 KBaud	4000000,00 Baud	1900	0
8	10000 Kbaud	10000000,00 Baud	A00	0
9	20000 Kbaud	20000000,00 Baud	500	0
10	33000 Kbaud	33032258,06 Baud	307	0,1

34.4.8 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 238*:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 238. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = 00’), 1 Stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = 01’), 1 Stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = ’10’), 1 Stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = 00’), 2 Stop bit	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01’), 2 Stop bit	1.82%	2.56%	3.90%	4.42%
7 bits (M = ’10’), 2 Stop bit	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 238 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = ‘00’ (11-bit times when M = ’01’ or 9- bit times when M = ‘10’).

34.4.9 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise Mute mode might remain active.*

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in LPUART_ISR register is set to ‘1’. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

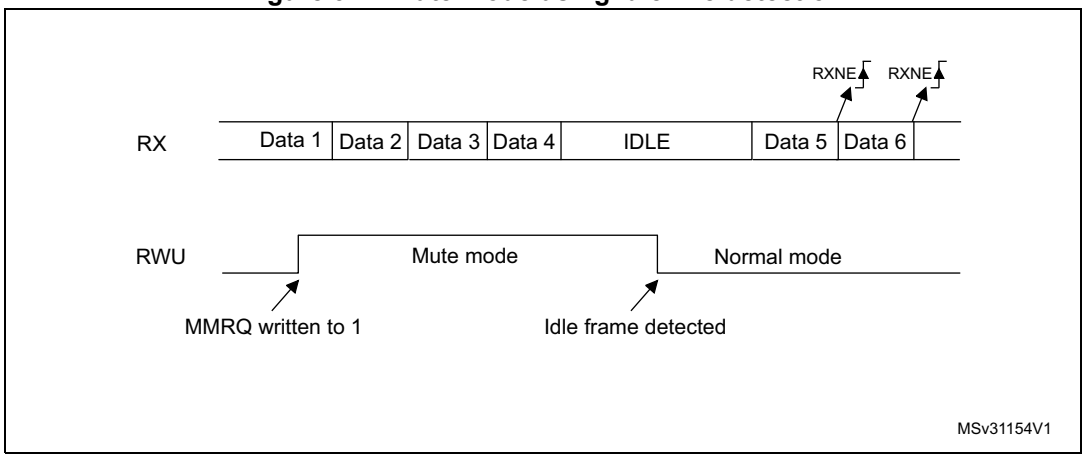
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The LPUART enters Mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of Mute mode behavior using Idle line detection is given in *Figure 324*.

Figure 324. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, the Mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’ otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters Mute mode.

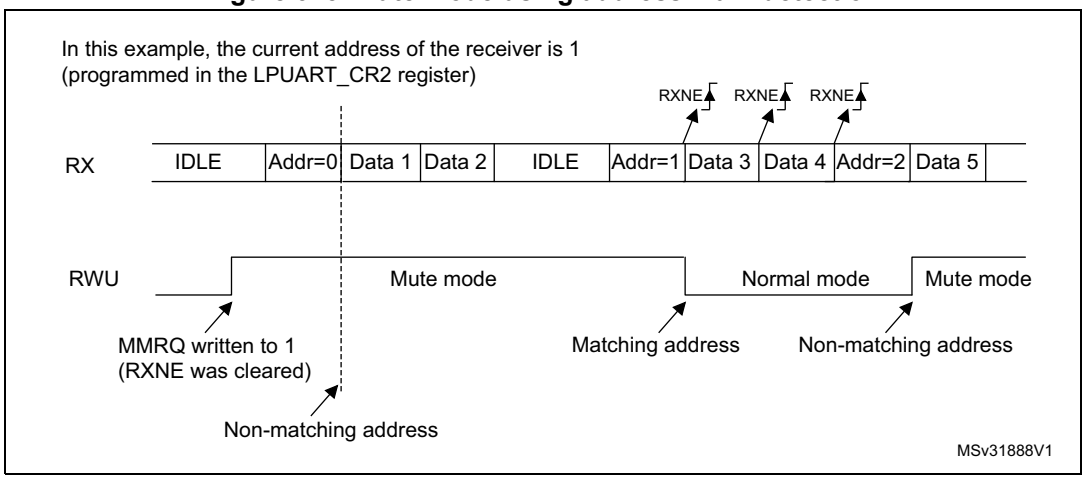
The LPUART also enters Mute mode when the MMRQ bit is written to ‘1’. The RWU bit is also automatically set in this case.

The LPUART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in Mute mode.

An example of Mute mode behavior using address mark detection is given in *Figure 325*.

Figure 325. Mute mode using address mark detection



34.4.10 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 239*.

Table 239: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data equal 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data equal 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

34.4.11 LPUART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the LPUART_CR2 register,
- SCEN and IREN bits in the LPUART_CR3 register.

The LPUART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

34.4.12 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 33.4: USART implementation on page 1017 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 33.5.6. To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

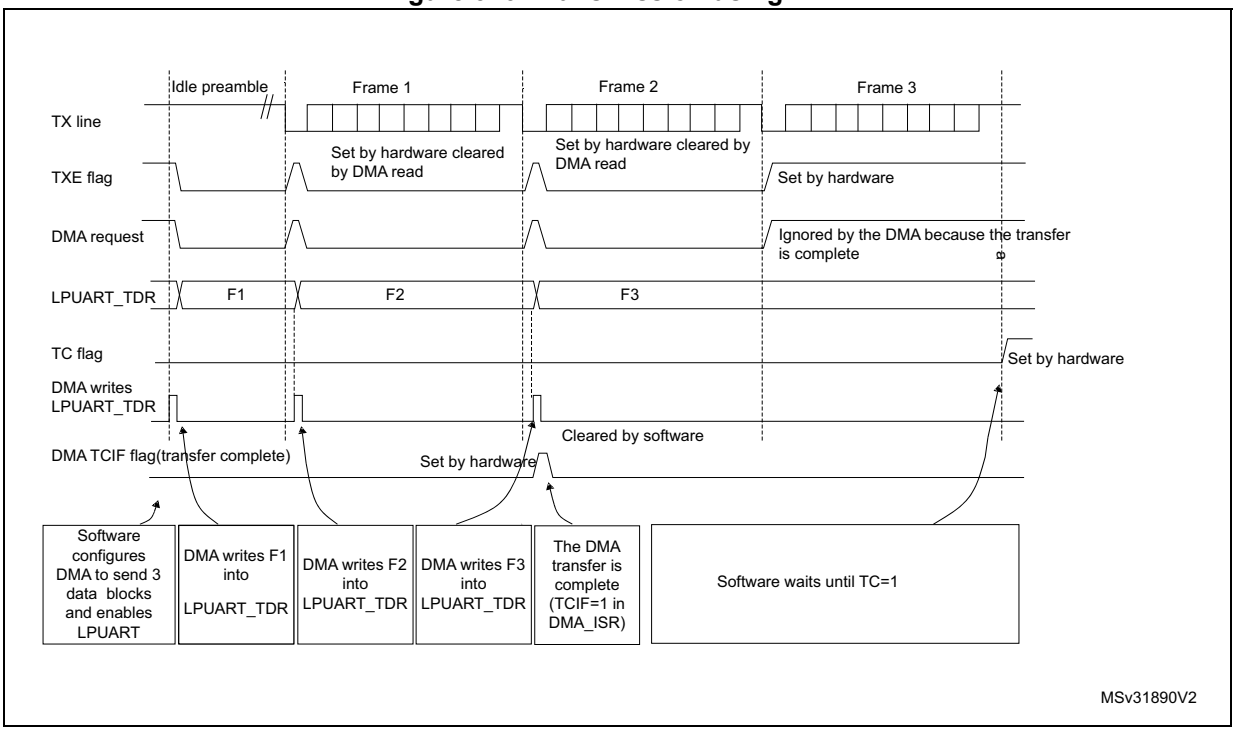
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the LPUART communication is complete. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 326. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

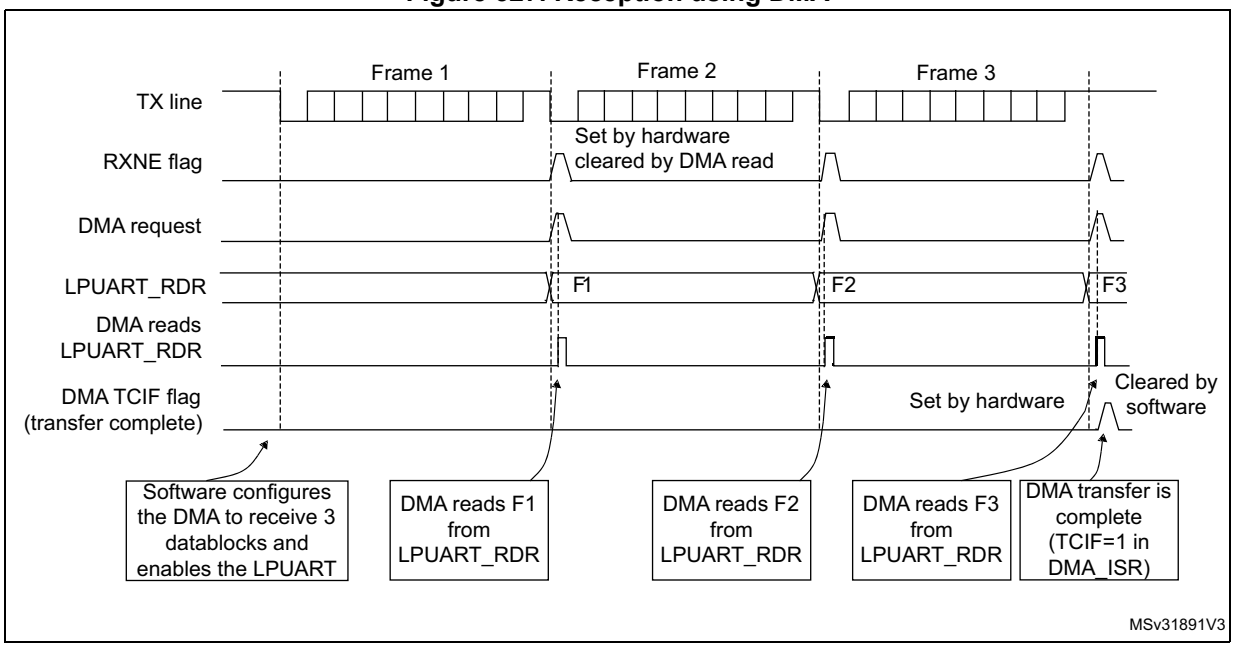
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller (DMA)* section) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 327. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

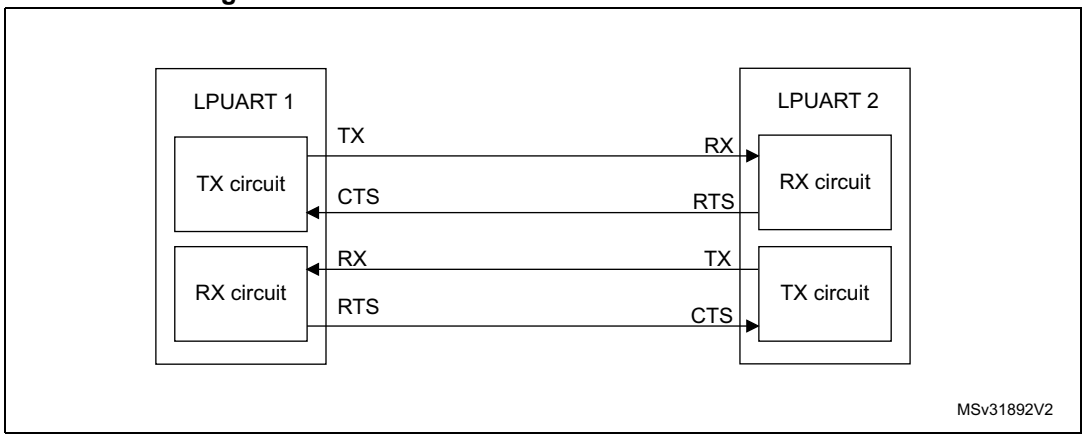
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt

enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

34.4.13 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The *Figure 314* shows how to connect 2 devices in this mode:

Figure 328. Hardware flow control between 2 LPUARTs

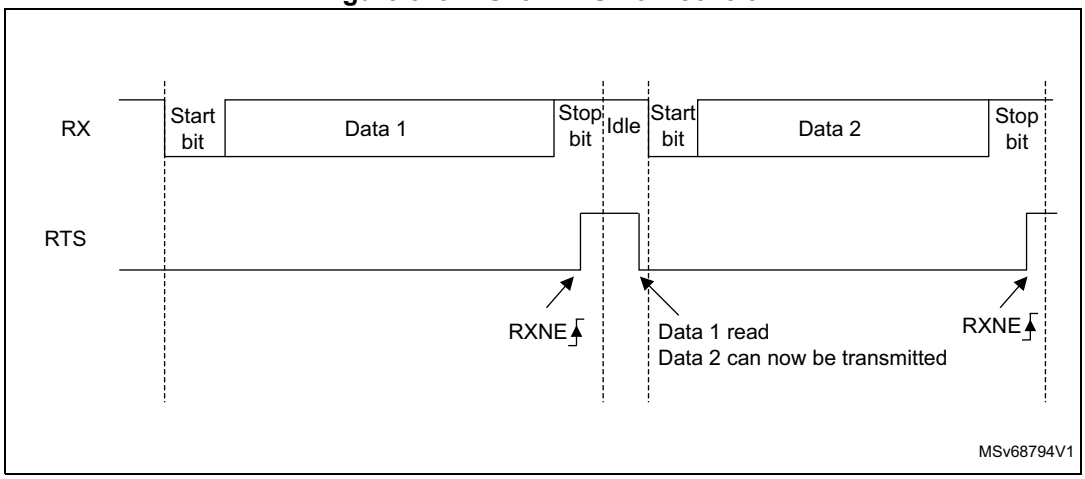


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 329* shows an example of communication with RTS flow control enabled.

Figure 329. RS232 RTS flow control



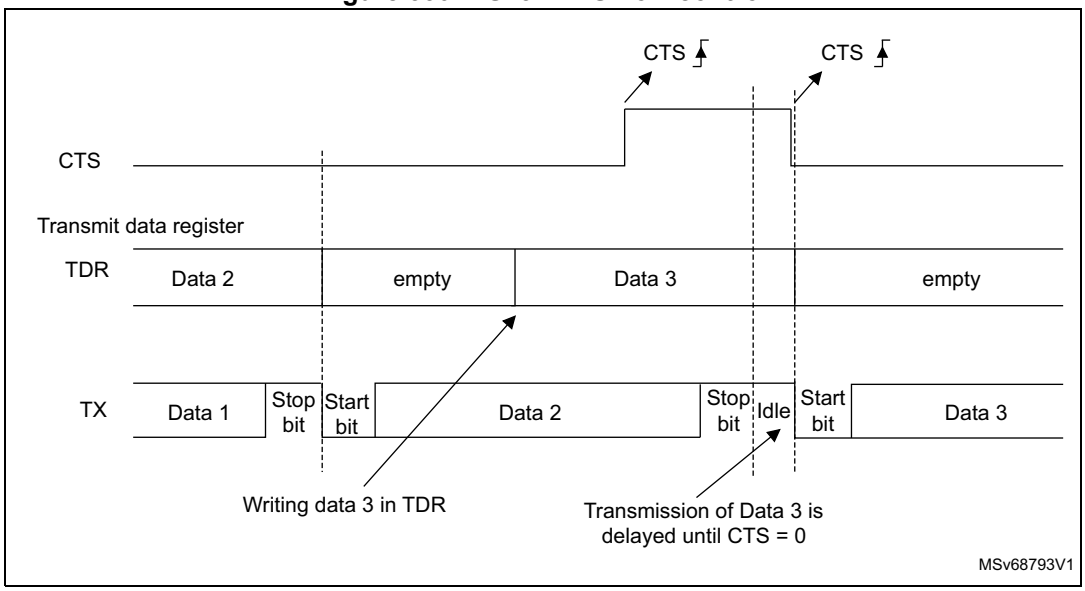
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 330* shows an example of communication with CTS flow control enabled.

Figure 330. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source (f_{CK}) cycles:

- The Driver enable assertion time equals
  - (1 + (DEAT x P)) x f_{CK}, if P # 0
  - (1 + DEAT) x f_{CK} , if P = 0
- The Driver enable deassertion time equals
  - (1 + (DEDT x P)) x f_{CK}, if P # 0
  - (1 + DEDT) x f_{CK}, if P = 0

where P = BRR[20:11]

34.4.14 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled.
The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set.
When the lpuart_pclk is gated, the LPUART provides a wakeup interrupt (lpuart_wkup) if a specific action requiring the activation of the lpuart_pclk clock is needed:

- If FIFO mode is disabled

  luart_pclk clock has to be activated to empty the LPUART data register.

  In this case, the lpuart_wkup interrupt source is the RXNE set to ‘1’. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled

  luart_pclk clock has to be activated
  - to fill the TXFIFO
  - or to empty the RXFIFO

  In this case, the lpuart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set .
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode enables doing as many LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific lpuart_wkup interrupt may be selected through the WUS bitfields.

When the wakeup event is detected, the WUF flag is set by hardware and lpuart_wkup interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled upon exit from low-power mode.

When FIFO is enabled, the wakeup from low-power mode on address match is only possible when Mute mode is enabled.

Using Mute mode with low-power mode

If the LPUART is put into Mute mode before entering low-power mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, Mute mode is used with wakeup from low-power mode without any constraints (i.e.the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wakeup from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, the lpuart_ker_ck is switched OFF again, the MCU is not waken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of wakeup event programmed to “address match detection” and FIFO management disabled.

Figure 331 shows the behavior when the wakeup event is verified.

Figure 331. Wakeup event verified (wakeup event = address match, FIFO disabled)

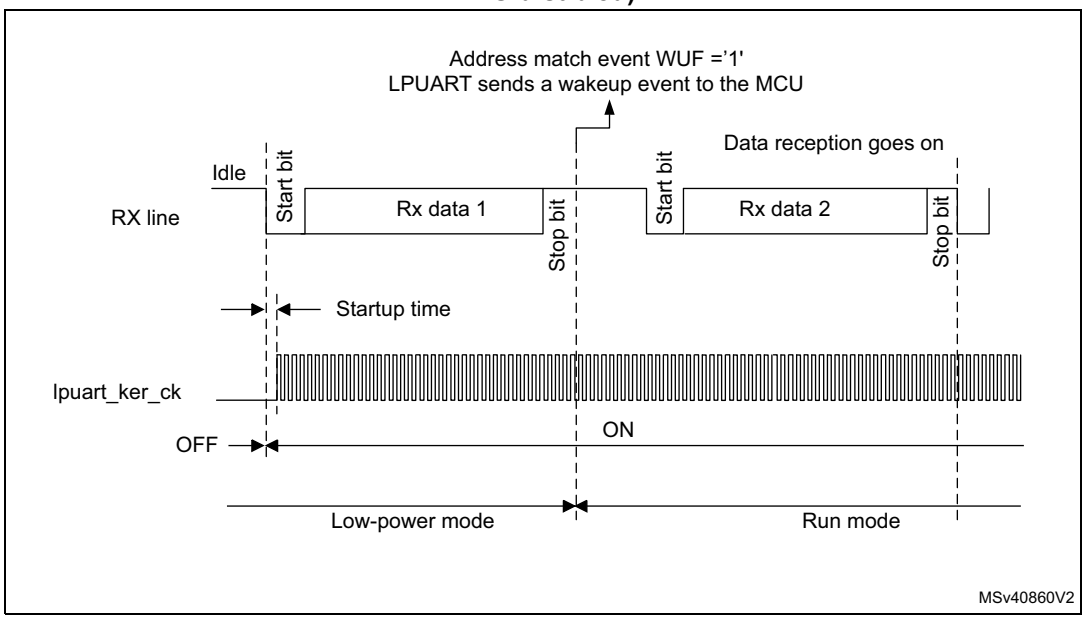
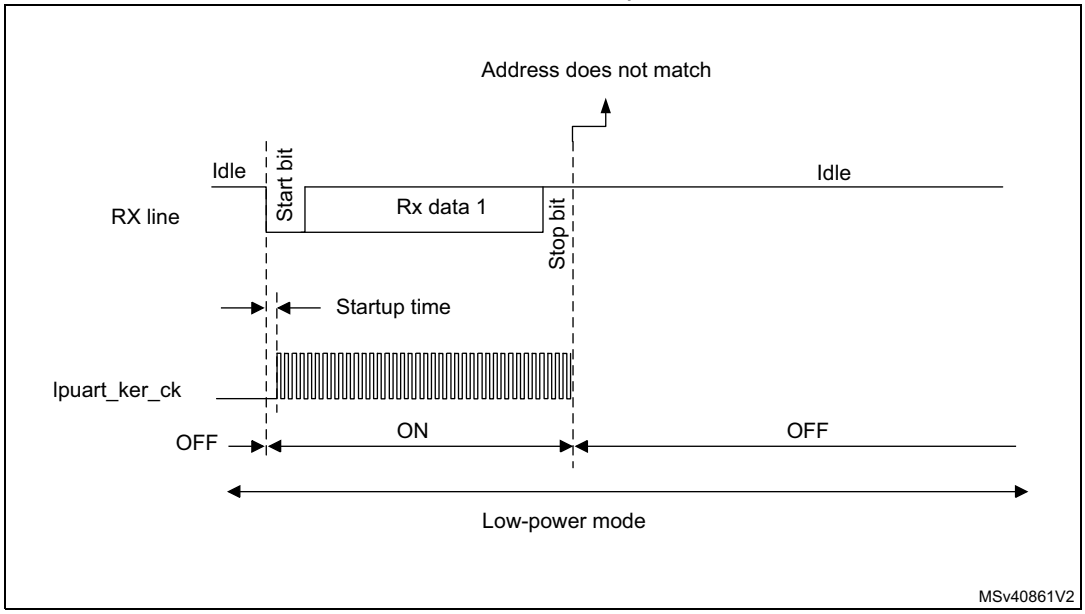


Figure 332 shows the behavior when the wakeup event is not verified.

Figure 332. Wakeup event not verified (wakeup event = address match, FIFO disabled)



Note: The above figures are valid when address match or any received frame is used as wakeup event. In the case the wakeup event is the start bit detection, the LPUART sends the wakeup event to the MCU at the end of the start bit.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 34.4.8: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 238: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% - 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wakeup correctly from low-power mode is: 1/11.32 µs = 88.36 kbaud.

34.5 LPUART in low-power modes

Table 240. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 34.3: LPUART implementation* to know if the wakeup from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

34.6 LPUART interrupts

Refer to *Table 241* for a detailed description of all LPUART interrupt requests.

Table 241. LPUART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(3)		Write 1 in ORECF		No	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Wakeup from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the Wakeup from Stop mode feature. Refer to *Section 34.3: LPUART implementation* for the list of supported Stop modes.
2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

34.7 LPUART registers

Refer to *Section 1.2 on page 55* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

34.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF = 1 in the LPUART_ISR register

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE = 1 in the LPUART_ISR register

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = ‘00’: 1 Start bit, 8 Data bits, n Stop bit

M[1:0] = ‘01’: 1 Start bit, 9 Data bits, n Stop bit

M[1:0] = ‘10’: 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 33.5.20: RS232 Hardware flow control and RS485 Driver Enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 34.4.13: RS232 Hardware flow control and RS485 Driver Enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE/RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.

When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

34.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = ‘00’: 1 Start bit, 8 Data bits, n Stop bit

M[1:0] = ‘01’: 1 Start bit, 9 Data bits, n Stop bit

M[1:0] = ‘10’: 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 33.5.20: RS232 Hardware flow control and RS485 Driver Enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 34.4.13: RS232 Hardware flow control and RS485 Driver Enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE/RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.

When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

34.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in Mute mode or a character code to be recognized in low-power or Run mode:

- In Mute mode: they are used in multiprocessor communication to wakeup from Mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wakeup from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with Mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)
1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} = 1/idle, Gnd = 0/mark)
1: TX pin signal values are inverted (V_{DD} = 0/mark, Gnd = 1/idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} = 1/idle, Gnd = 0/mark)
1: RX pin signal values are inverted (V_{DD} = 0/mark, Gnd = 1/idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: STOP bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

34.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS[1:0]		Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Remaining combinations: Reserved.

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Remaining combinations: Reserved.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wakeup from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01:Reserved.
10: WUF active on Start bit detection
11: WUF active on RXNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on Reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables checking the communication flow w/o reading the data.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of Single-wire Half-duplex mode
0: Half duplex mode is not selected
1: Half duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

34.7.5 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]**: LPUART baud rate

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≧ 0x300 and LPUART_BRR is 20 bits, a care should be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].

34.7.6 LPUART request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

34.7.7 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG in LPUART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in RXFTCFG in LPUART_CR3 register i.e. the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not full

1: RXFIFO is full

Bit 23 **TXFE**: TXFIFO empty

This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty

1: TXFIFO is empty

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in Active mode

1: Receiver in Mute mode

Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in LPUART_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception on going

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXFF is set. An interrupt is generated if TCIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART_ICR register or by a write to the LPUART_TDR register.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE = 1 or EIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

34.7.8 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in LPUART_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception on going

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty/TXFIFO not full

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit =1 in the LPUART_CR1 register.

0: Data register full

1: Data register not full

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART_ICR register or by a write to the LPUART_TDR register.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by reading from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE = 1 or EIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

34.7.9 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

34.7.10 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 319*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

34.7.11 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 319*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

34.7.12 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaining combinations: Reserved.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.

34.7.13 LPUART register map

The table below gives the LPUART register map and reset values.

Table 242. LPUART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]					Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	LPUART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]					Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS [1:0]		Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0	0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFF	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0		0	1	0	0	0	0	0	0	0						0	0		1	1	0	0	0	0	0	0
0x1C	LPUART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0	0	0	0	0	0						0	0		1	1	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 242. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	LPUART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

35 Serial peripheral interface / integrated interchip sound (SPI/I2S)

35.1 Introduction

The SPI/I²S interface can be used to communicate with external devices using the SPI protocol or the I^2S audio protocol. SPI or I^2S mode is selectable by software. SPI Motorola mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

The integrated interchip sound (I^2S) protocol is also a synchronous serial communication interface. It can operate in slave or master mode with half-duplex communication. It can address four different audio standards including the Philips I^2S standard, the MSB- and LSB-justified standards and the PCM standard.

35.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4 to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to $f_{PCLK}/2$
- Slave mode frequency up to $f_{PCLK}/2$.
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- Enhanced TI and NSS pulse modes support

35.3 I2S main features

- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode, overrun flag in reception mode (master and slave) and Frame Error Flag in reception and transmitter mode (slave only)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock can be output to drive an external audio component. The ratio is fixed at 256 × f_s for all I2S modes, and to 128 x f_s for all PCM modes (where f_s is the audio sampling frequency).

35.4 SPI/I2S implementation

The following table describes all the SPI instances and their features embedded in the devices.

Table 243. STM32WLEx SPI and SPI/I2S implementation(1)

SPI Features	SPI1	SPI2S2	SUBGHZSPI
Enhanced NSSP and TI modes	Yes	Yes	Yes
Hardware CRC calculation	Yes	Yes	No
I^2S support	No	Yes	No
Data size configurable	from 4 to 16-bit	from 4 to 16-bit	from 4 to 16-bit
Rx/Tx FIFO size	32-bit	32-bit	32-bit

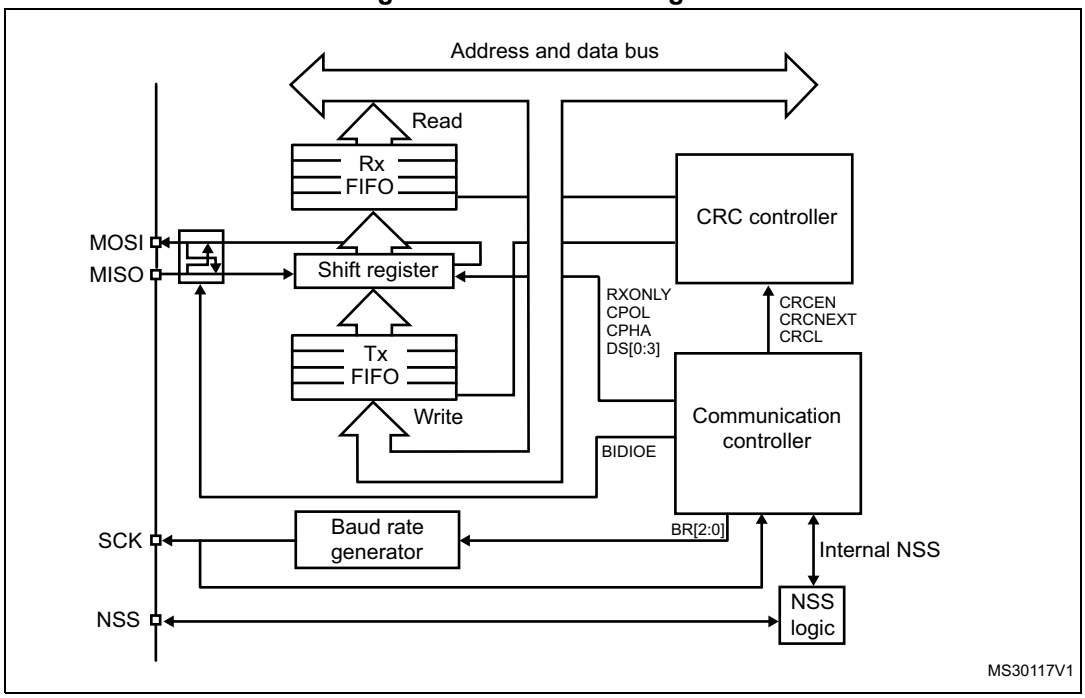
1. The SPI1 and SPI2S2 instances are general purpose type while the SUBGHZSPI instance is dedicated for Sub-GHz radio control exclusively.

35.5 SPI functional description

35.5.1 General description

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram *Figure 333*.

Figure 333. SPI block diagram



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

  See *Section 35.5.5: Slave select (NSS) pin management* for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

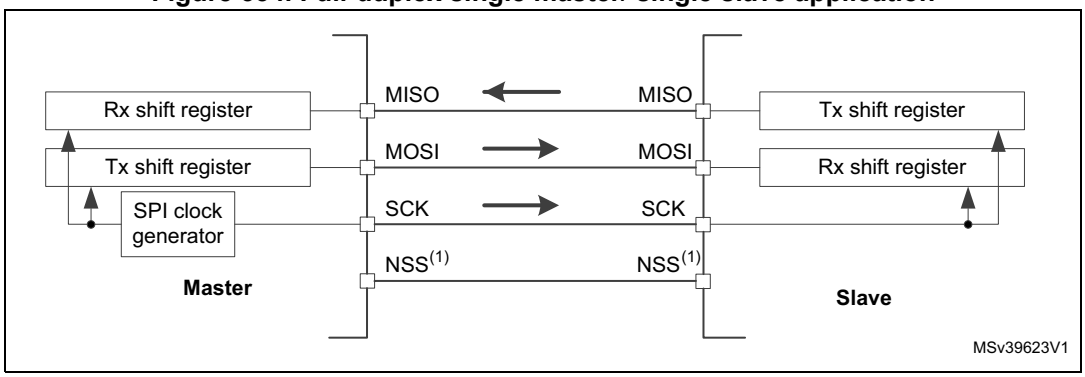
35.5.2 Communications between one master and one slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management). Communication is always initiated by the master.

Full-duplex communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 334. Full-duplex single master/ single slave application

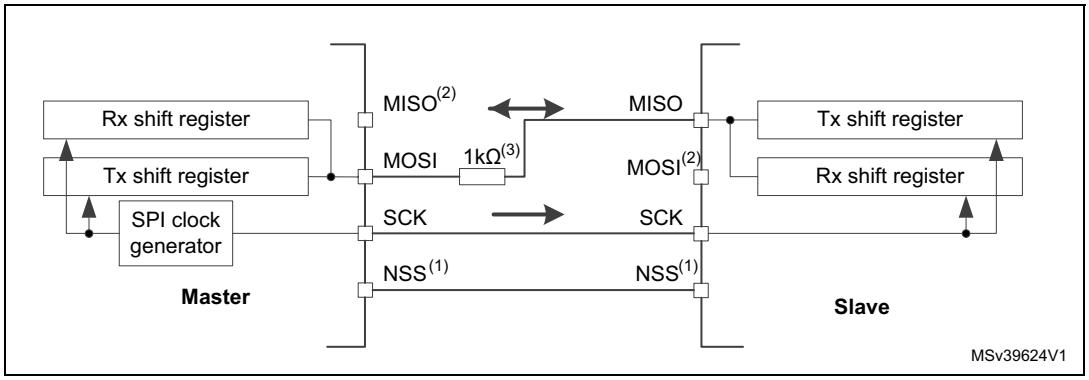


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 35.5.5: Slave select (NSS) pin management*.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx_CR1 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

Figure 335. Half-duplex single master/ single slave application



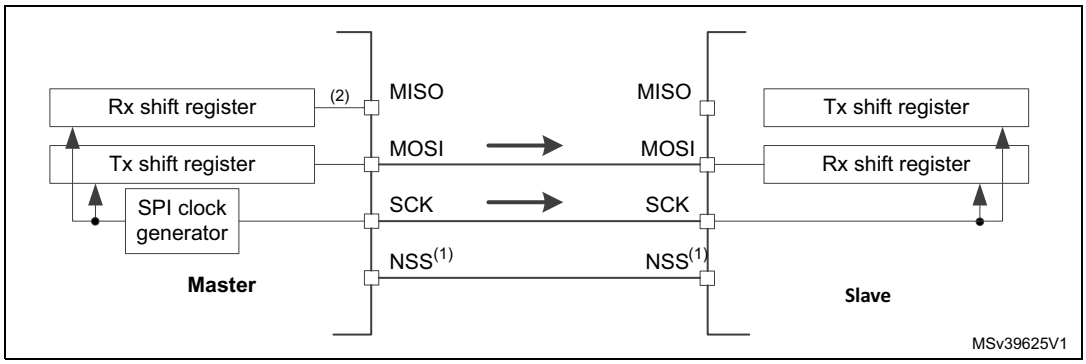
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 35.5.5: Slave select (NSS) pin management*.
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.
3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectionnal mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communication data). Both nodes then fight while providing opposite output levels on the common line temporary till next node changes its direction settings correspondingly, too. It is suggested to insert a serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx_CR1 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1)**: The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *35.5.5: Slave select (NSS) pin management*). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

Figure 336. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)



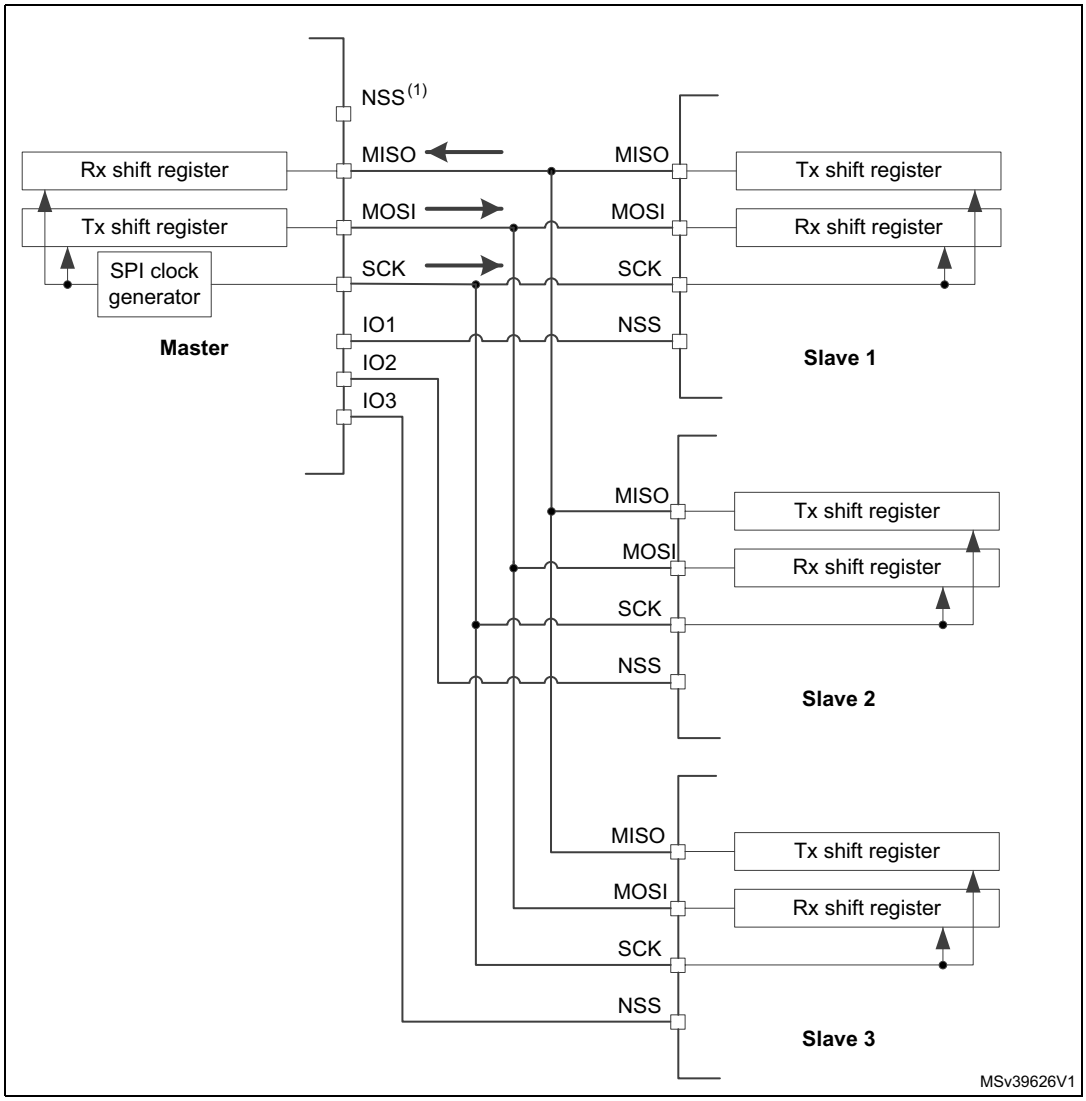
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 35.5.5: Slave select (NSS) pin management*.
2. An accidental input information is captured at the input of transmitter Rx shift register. All the events associated with the transmitter receive flow must be ignored in standard transmit only mode (e.g. OVR flag).
3. In this configuration, both the MISO pins can be used as GPIOs.

Note: Any simplex communication can be alternatively replaced by a variant of the half-duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).

35.5.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see *Figure 337*). The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input. When this is done, a standard master and dedicated slave communication is established.

Figure 337. Master and three independent slaves



1. NSS pin is not used on master side at this configuration. It has to be managed internally (SSM=1, SSI=1) to prevent any MODF error.
2. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see I/O alternate function input/output section (GPIO)).

35.5.4 Multi-master communication

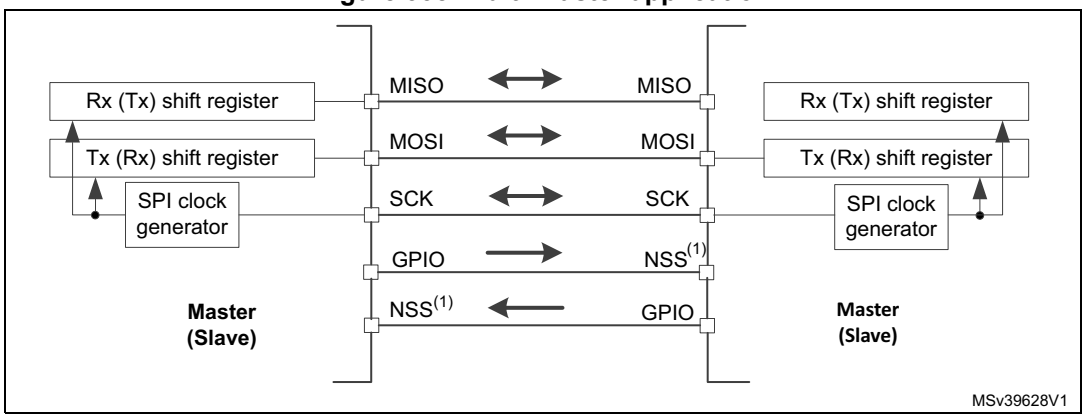
Unless SPI bus is not designed for a multi-master capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

Figure 338. Multi-master application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.

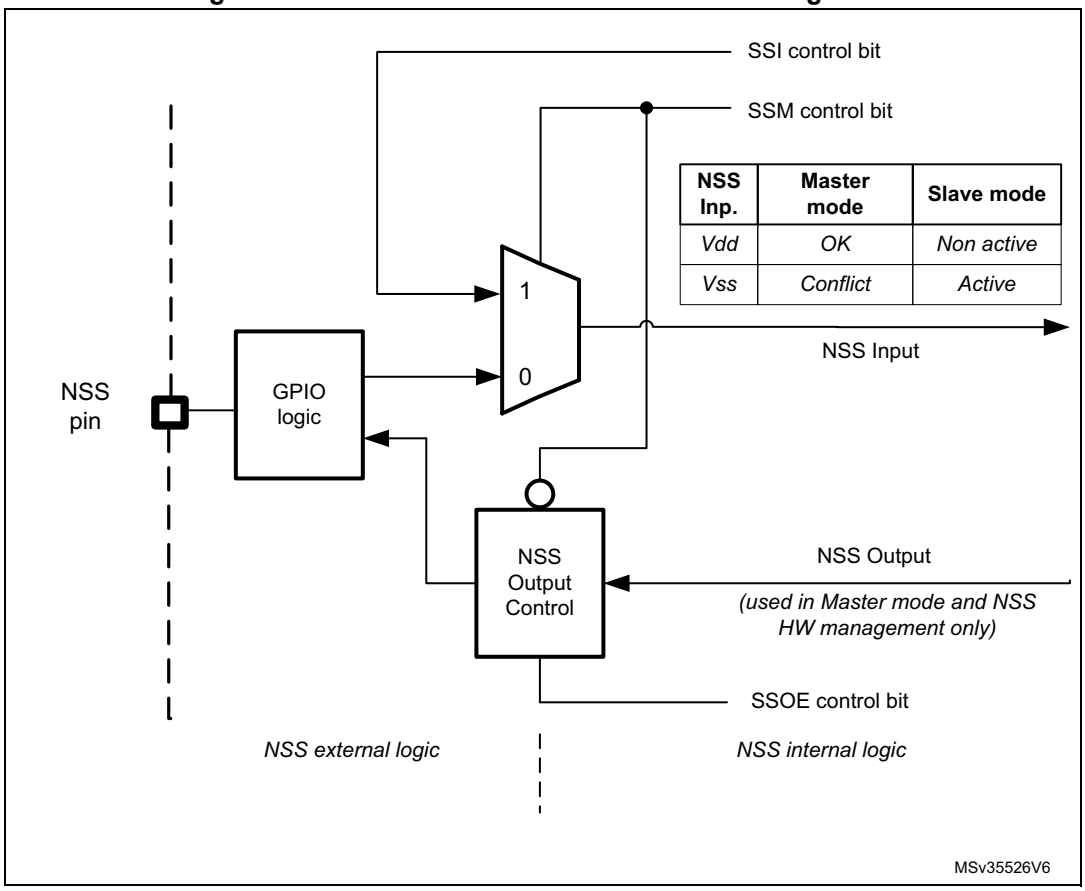
35.5.5 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

- **Software NSS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx_CR1).
  - **NSS output enable (SSM=0,SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
  - **NSS output disable (SSM=0, SSOE = 0)**: if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 339. Hardware/software slave select management



35.5.6 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

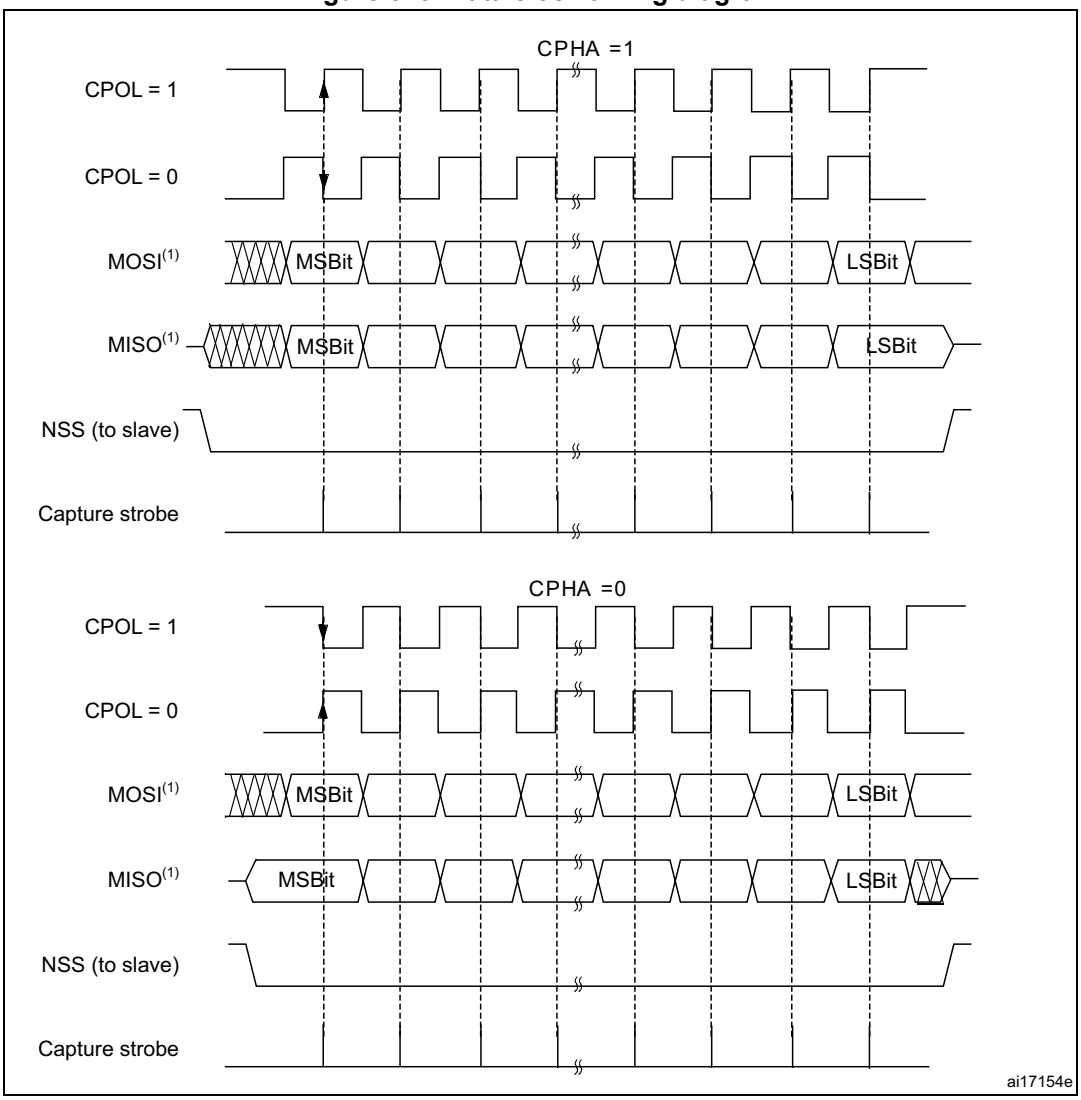
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 340, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPIx_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

Figure 340. Data clock timing diagram

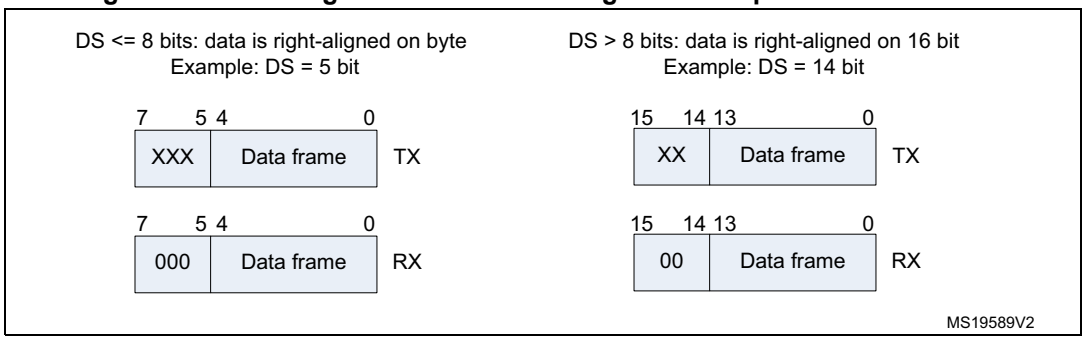


1. The order of data bits depends on LSBFIRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit. The data frame size is chosen by using the DS bits. It can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception. Whatever the selected data frame size, read access to the FIFO must be aligned with the FRXTH level. When the SPIx_DR register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte) or a half-word (see *Figure 341*). During communication, only bits within the data frame are clocked and transferred.

Figure 341. Data alignment when data length is not equal to 8-bit or 16-bit



Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 8-bit data frame size.

35.5.7 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated sections. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI_CR1 register:
   a) Configure the serial clock baud rate using the BR[2:0] bits (Note: 4).
   b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). (Note: 2 - except the case when CRC is enabled at TI mode).
   c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE cannot be set at the same time).
   d) Configure the LSBFIRST bit to define the frame format (Note: 2).
   e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
   f) Configure SSM and SSI (Notes: 2 & 3).
   g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI_CR2 register:
   a) Configure the DS[3:0] bits to select the data length for the transfer.
   b) Configure SSOE (Notes: 1 & 2 & 3).
   c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
   d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CHPA and TI bits cleared in NSSP mode).
   e) Configure the FRXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx_DR register.
   f) Initialize LDMA_TX and LDMA_RX bits if DMA is used in packed mode.
4. Write to SPI_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

Note: (1) Step is not required in slave mode.

(2) Step is not required in TI mode.

(3) Step is not required in NSSP mode.

(4) The step is not required in slave mode except slave working at TI mode

35.5.8 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full-duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated section.

35.5.9 Data transmission and reception procedures

RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see *Section 35.5.14: CRC calculation*).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see *Section 35.5.13: TI mode*).

A read access to the SPIx_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, RXFIFO can store up to four data frames, whereas TXFIFO can only store up to three when the data frame format is not greater than 8 bits. This difference prevents possible corruption of 3x 8-bit data frames already stored in the TXFIFO when software tries to write more data in 16-bit mode into TXFIFO. Both TXE and RXNE events can be polled or handled by interrupts. See *Figure 343* through *Figure 346*.

Another way to manage the data exchange is to use DMA (see *Communication using DMA (direct memory addressing)*).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at *Section 35.5.10: SPI status flags*). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half-duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see *Section 35.5.5: Slave select (NSS) pin management*).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full-duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to prevent some dummy byte exchange (refer to *Data packing* section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When

the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional “dummy” data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIiRST bits in the RCC_APBiRSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave, or
- When transactions’ streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

Note: If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.

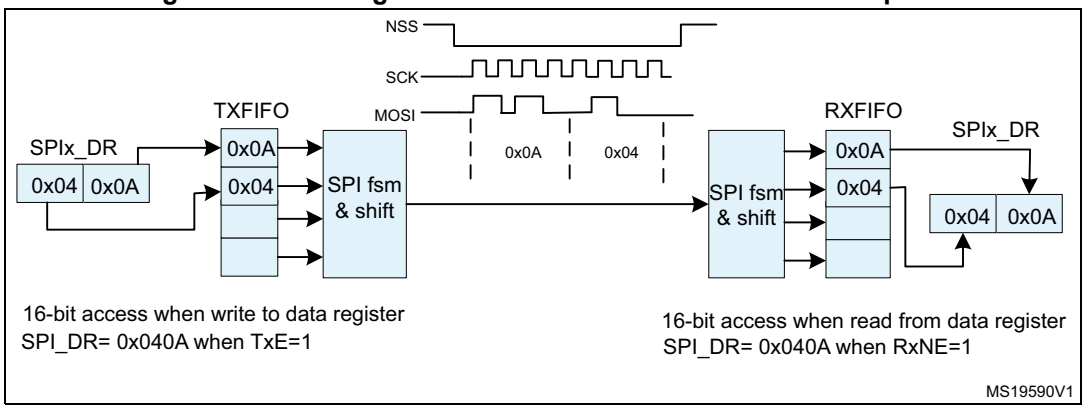
Data packing

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. *Figure 342* provides an example of data packing mode sequence handling. Two data frames are sent after the single 16-bit access the SPIx_DR register of the transmitter. This sequence can generate just one RXNE event in the receiver if the RXFIFO threshold is set to 16 bits (FRXTH=0). The receiver then has to access both data frames by a single 16-bit read of SPIx_DR as a response to this single RXNE event. The

RxFIFO threshold setting and the following read access must be always kept aligned at the receiver side, as data can be lost if it is not in line.

A specific problem appears if an odd number of such “fit into one byte” data frames must be handled. On the transmitter side, writing the last data frame of any odd sequence with an 8-bit access to SPIx_DR is enough. The receiver has to change the Rx_FIFO threshold level for the last data frame received in the odd sequence of frames in order to generate the RXNE event.

Figure 342. Packing data in FIFO for transmission and reception



1. In this example: Data size DS[3:0] is 4-bit configured, CPOL=0, CPHA=1 and LSBFIRST =0. The Data storage is always right aligned while the valid bits are performed on the bus only, the content of LSB byte goes first on the bus, the unused bits are not taken into account on the transmitter side and padded by zeros at the receiver side.

Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bit in the SPIx_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx_DR register.

See *Figure 343* through *Figure 346*.

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received is not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until FTLVL[1:0]=00 and then until BSY=0.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CR2 register, if DMA Rx is used.
2. Enable DMA streams for Tx and Rx in DMA registers, if the streams are used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CR2 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA streams for Tx and Rx in the DMA registers, if the streams are used.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CR2 register, if DMA Tx and/or DMA Rx are used.

Packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPIx_CR2 register) packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel. If the DMA channel PSIZE value is equal to 16-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. The DMA then automatically manages the write operations to the SPIx_DR register.

If data packing mode is used and the number of data to transfer is not a multiple of two, the LDMA_TX/LDMA_RX bits must be set. The SPI then considers only one data for the transmission or reception to serve the last DMA transfer (for more details refer to *Data packing on page 1168*.)

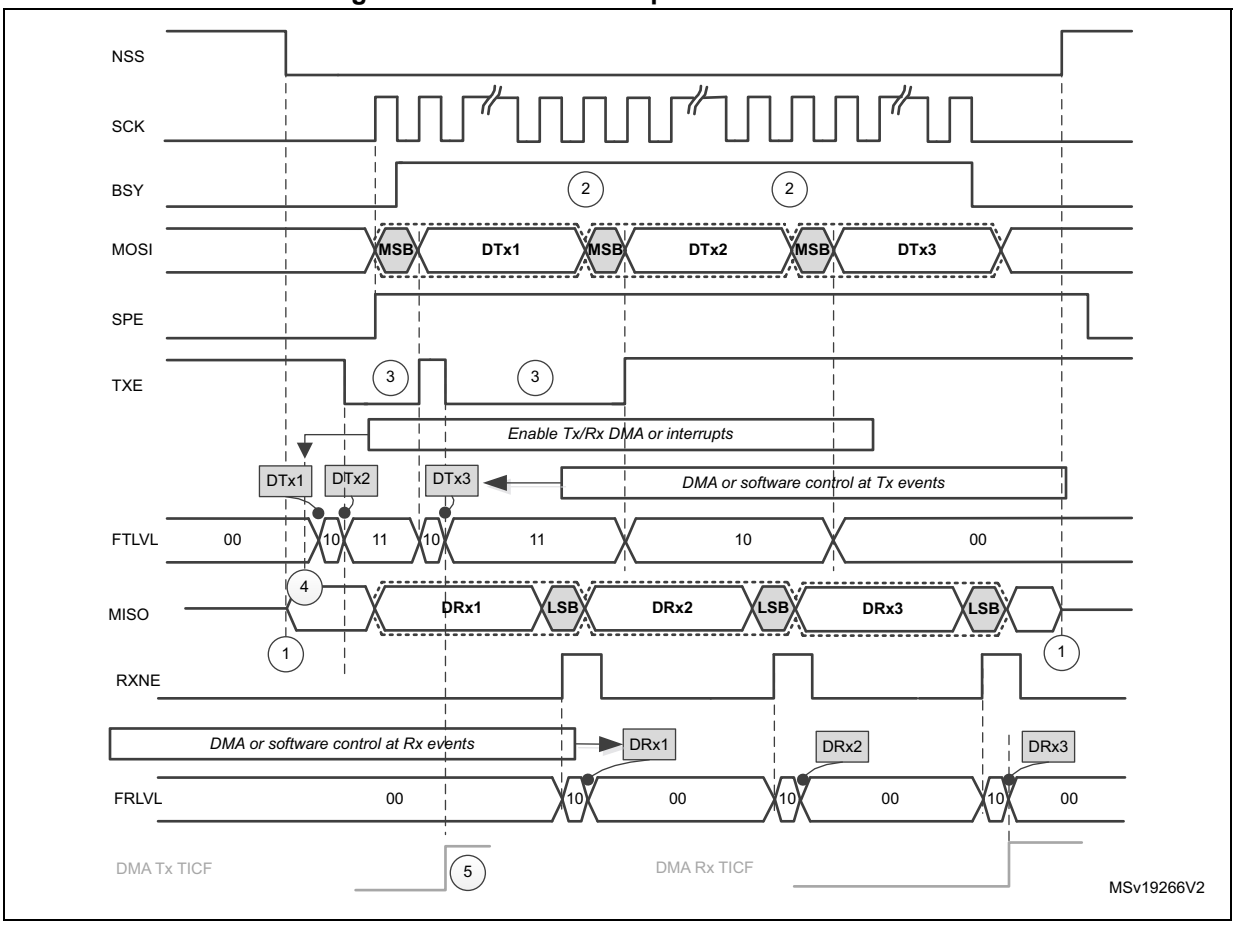
Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by polling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for *Figure 343 on page 1172* through *Figure 346 on page 1175*:

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.
   At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depends on GPIO setting exclusively.
2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.
3. The TXE signal is cleared only if TXFIFO is full.
4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.
5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.
6. The CRC value for a package is calculated continuously frame by frame in the SPIx_TXCRCR and SPIx_RXCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).
   While the CRC value calculated in SPIx_TXCRCR is simply sent out by transmitter, received CRC information is loaded into RxFIFO and then compared with the SPIx_RXCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of RxFIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).
7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the TxFIFO is ¾ full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the TxFIFO becomes ½ full. This frame is stored into TxFIFO with an 8-bit access either by software or automatically by DMA when LDMA_TX control is set.
8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA_RX is set.

Figure 343. Master full-duplex communication



Assumptions for master full-duplex communication example:

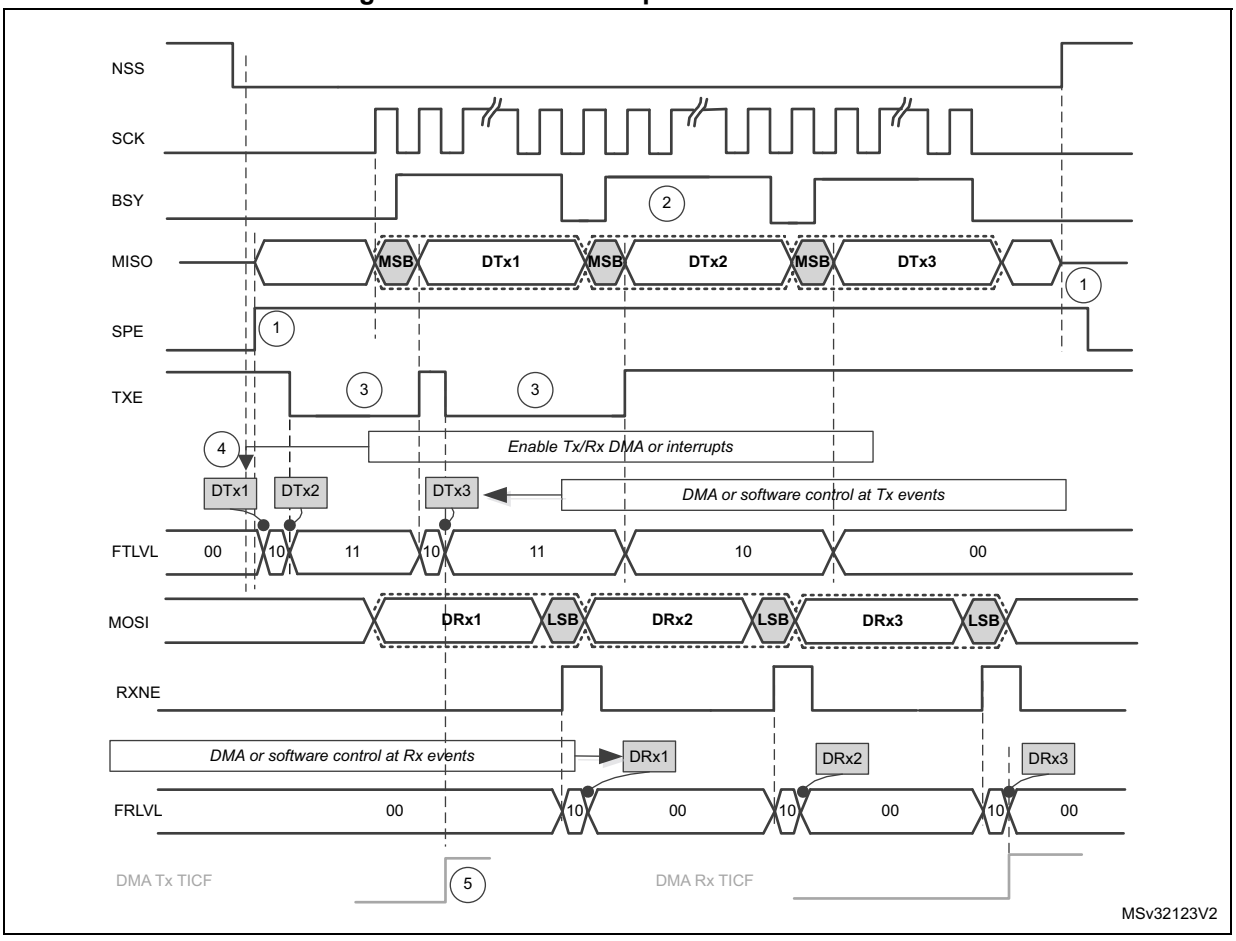
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *: Communication diagrams on page 1171* for details about common assumptions and notes.

Figure 344. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

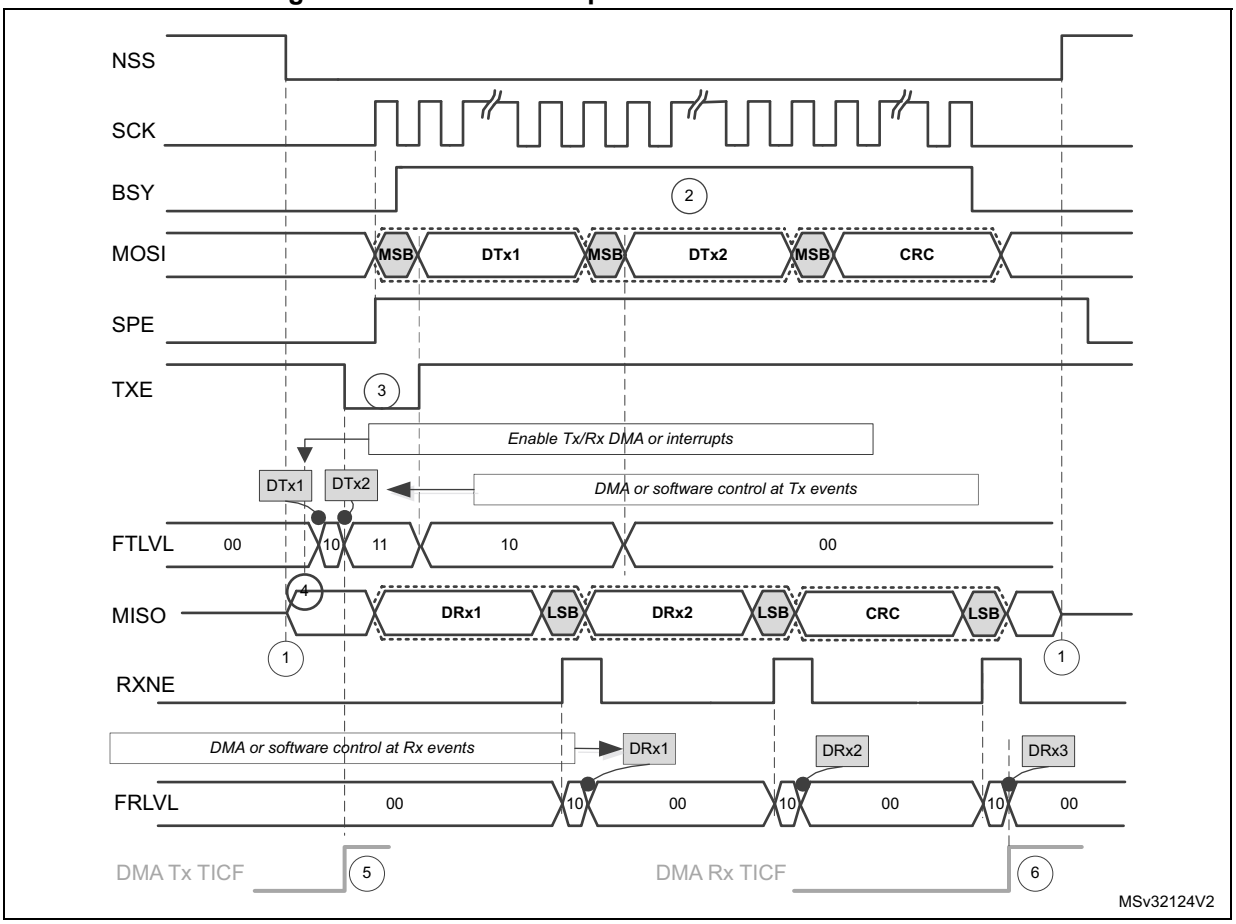
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *Communication diagrams on page 1171* for details about common assumptions and notes.

Figure 345. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

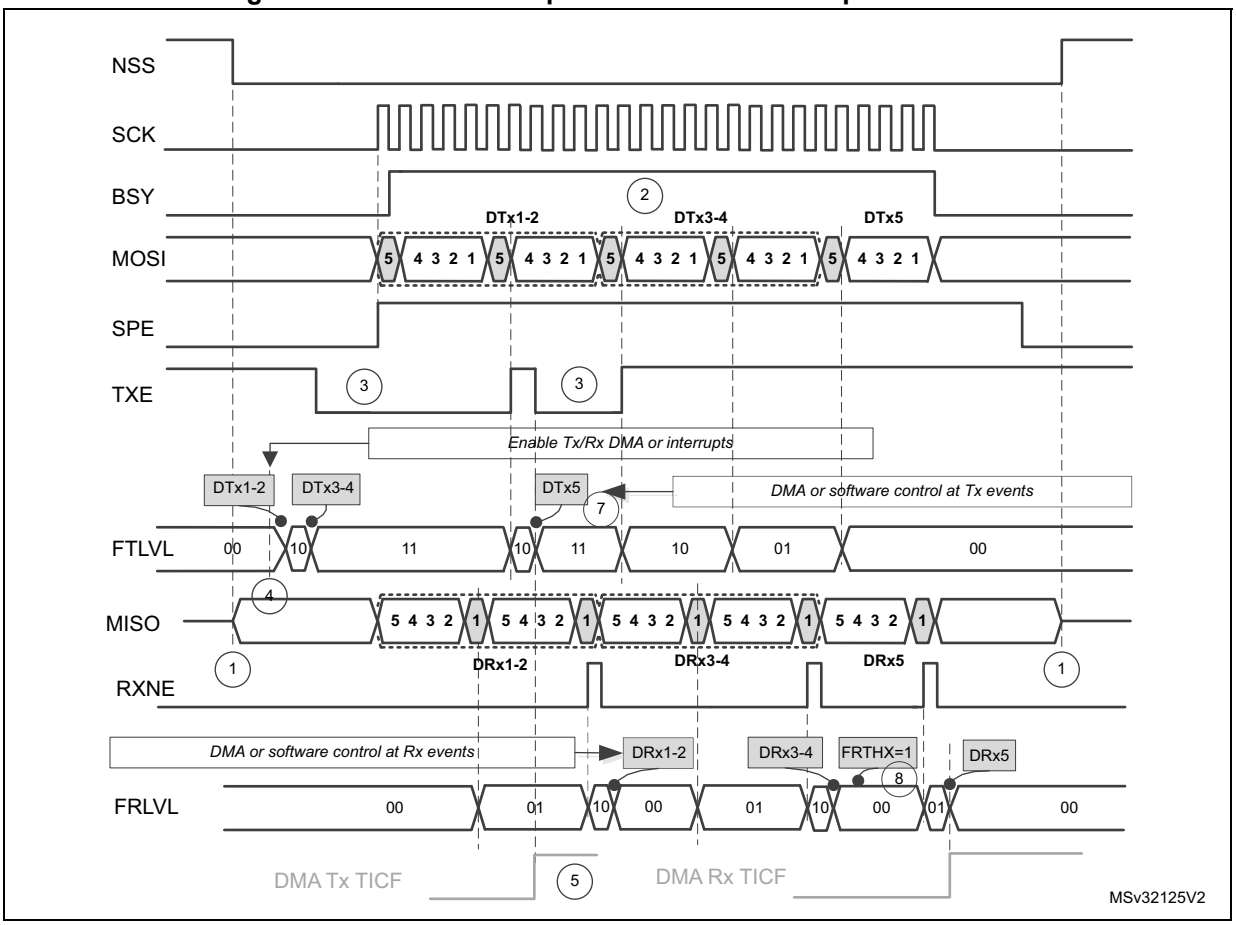
- Data size = 16 bit
- CRC enabled

If DMA is used:

- Number of Tx frames transacted by DMA is set to 2
- Number of Rx frames transacted by DMA is set to 3

See also *: Communication diagrams on page 1171* for details about common assumptions and notes.

Figure 346. Master full-duplex communication in packed mode



Assumptions for master full-duplex communication in packed mode example:

- Data size = 5 bit
- Read/write FIFO is performed mostly by 16-bit access
- FRXTH=0

If DMA is used:

- Number of Tx frames to be transacted by DMA is set to 3
- Number of Rx frames to be transacted by DMA is set to 3
- PSIZE for both Tx and Rx DMA channel is set to 16-bit
- LDMA_TX=1 and LDMA_RX=1

See also *: Communication diagrams on page 1171* for details about common assumptions and notes.

35.5.10 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx_CR2 register:

- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx_CR2 register is set.

The RXNE is cleared by hardware automatically when the above conditions are no longer true.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note: When the next transmission can be handled immediately by the master (e.g. if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.

35.5.11 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

Overrun flag (OVR)

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited e.g. the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see *Section 35.5.14: CRC calculation*).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI_DR register followed by a read access to the SPI_SR register.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx_SR register while the MODF bit is set.
2. Then write to the SPIx_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx_CR1 register is set. The CRCERR flag in the SPIx_SR register is set if the value received in the shift register does not match the receiver SPIx_RXCRCR value. The flag is cleared by the software.

TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

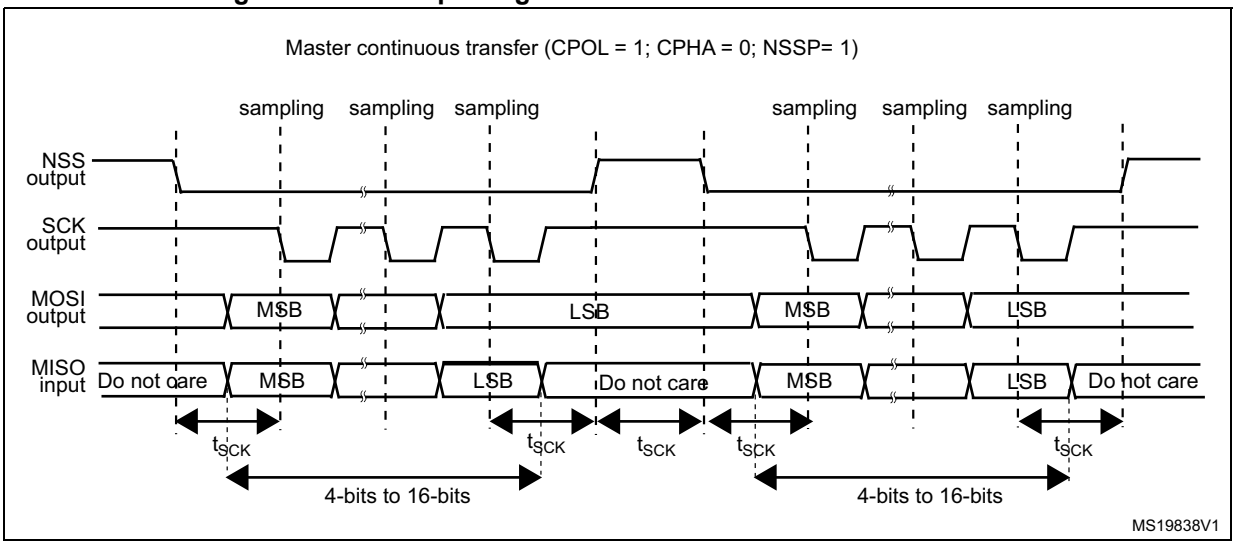
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

35.5.12 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

Figure 347 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 347. NSSP pulse generation in Motorola SPI master mode



Note: Similar behavior is encountered when CPOL = 0. In this case the sampling edge is the *rising* edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

35.5.13 TI mode

TI protocol in master mode

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx_CR1 and SPIx_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see *Figure 348*). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ ($t_{release}$) depends on internal resynchronization and on the

baud rate value set in through the BR[2:0] bits in the SPIx_CR1 register. It is given by the formula:

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

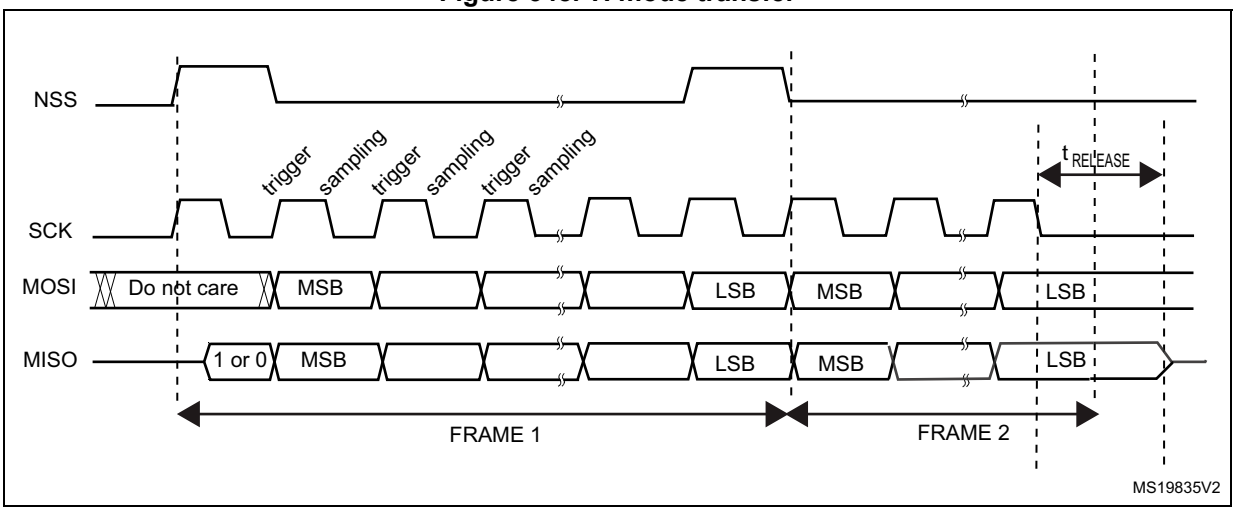
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

If the data size is equal to 4-bits or 5-bits, the master in full-duplex mode or transmit-only mode uses a protocol with one more dummy data bit added after LSB. TI NSS pulse is generated above this dummy bit clock cycle instead of the LSB in each period.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

Figure 348: TI mode transfer shows the SPI communication waveforms when TI mode is selected.

Figure 348. TI mode transfer



35.5.14 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

Note: The polynomial value should only be odd. No even values are supported.

CRC transfer managed by CPU

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx_DR register. Then CRCNEXT bit has to be set in the SPIx_CR1 register to indicate that the CRC frame transaction follows after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPIx_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx_DR register in order to clear the RXNE flag.

CRC transfer managed by DMA

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full-duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

DMA_RX = Numb_of_data + 2

In receive only mode, the DMA reception channel counter should contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA_RX bit needs managing if the number of data is odd.

Resetting the SPIx_TXCRC and SPIx_RXCRC values

The SPIx_TXCRC and SPIx_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).

If the SPI is disabled during a communication the following sequence must be followed:

1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

Note: When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low during transaction of the CRC phase once the CRCNEXT signal is released. That is why the CRC calculation cannot be used at NSS Pulse mode when NSS hardware mode should be applied at slave normally.

At TI mode, despite the fact that clock phase and clock polarity setting is fixed and independent on SPIx_CR1 register, the corresponding setting CPOL=0 CPHA=1 has to be kept at the SPIx_CR1 register anyway if CRC is applied. In addition, the CRC calculation has to be reset between sessions by SPI disable sequence with re-enable the CRCEN bit described above at both master and slave side, else CRC calculation can be corrupted at this specific mode.

35.6 SPI interrupts

During SPI communication an interrupt can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error
- CRC protocol error

Interrupts can be enabled and disabled separately.

Table 244. SPI interrupt requests

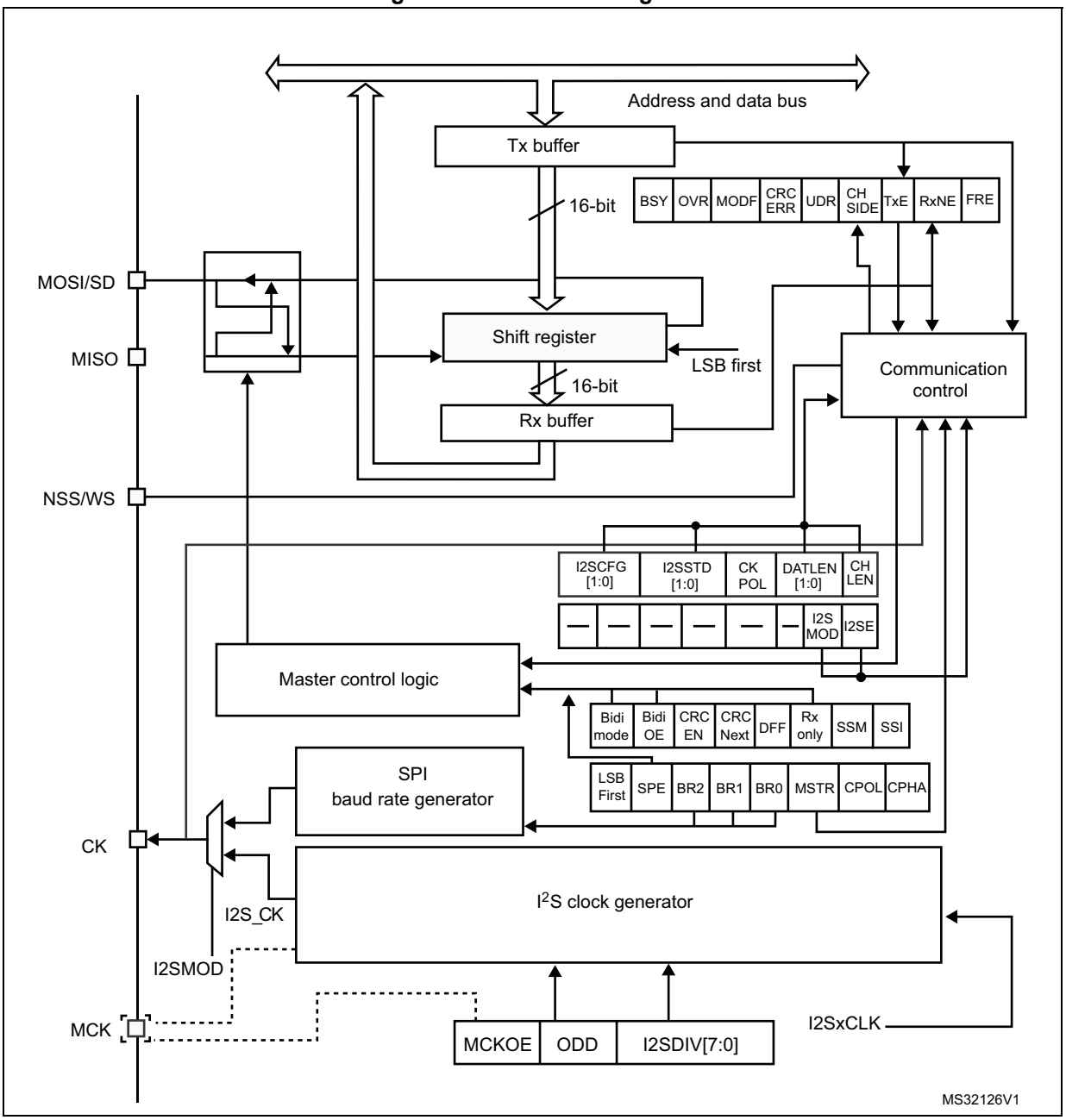
Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	
CRC protocol error	CRCERR	

35.7 I2S functional description

35.7.1 I2S general description

The block diagram of the I2S is shown in *Figure 349*.

Figure 349. I2S block diagram



1. MCK is mapped on the MISO pin.

The SPI can function as an audio I2S interface when the I2S capability is enabled (by setting the I2SMOD bit in the SPIx_I2SCFGR register). This interface mainly uses the same pins, flags and interrupts as the SPI.

The I2S shares three common pins with the SPI:

- SD: Serial Data (mapped on the MOSI pin) to transmit or receive the two time-multiplexed data channels (in half-duplex mode only).
- WS: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I2S is configured in master mode (and when the MCKOE bit in the SPIx_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to 256 × f_S for all I2S modes, and to 128 x f_S for all PCM modes, where f_S is the audio sampling frequency.

The I2S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I^2S mode. One is linked to the clock generator configuration SPIx_I2SPR and the other one is a generic I2S configuration register SPIx_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPIx_CR1 register and all CRC registers are not used in the I^2S mode. Likewise, the SSOE bit in the SPIx_CR2 register and the MODF and CRCERR bits in the SPIx_SR are not used.

The I2S uses the same SPI register for data transfer (SPIx_DR) in 16-bit wide mode.

35.7.2 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for transmission or reception. So, it is up to the software to write into the data register the appropriate value corresponding to each channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPIx_SR register. Channel left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol).

Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in a 16-bit frame
- 16-bit data packed in a 32-bit frame
- 24-bit data packed in a 32-bit frame
- 32-bit data packed in a 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPIx_DR register or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 non-significant bits are extended to 32 bits with 0-bits (by hardware).

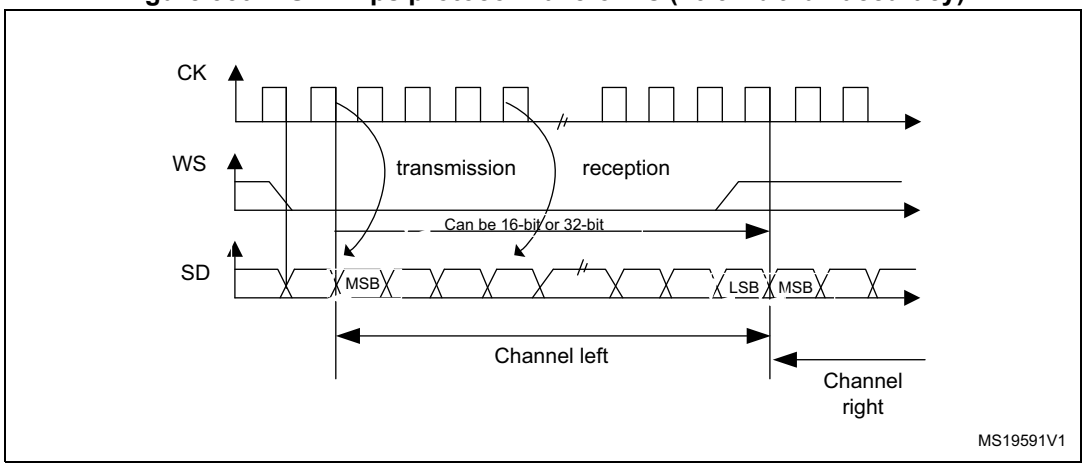
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I^{2}S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx_I2SCFGR register.

I^{2}S Philips standard

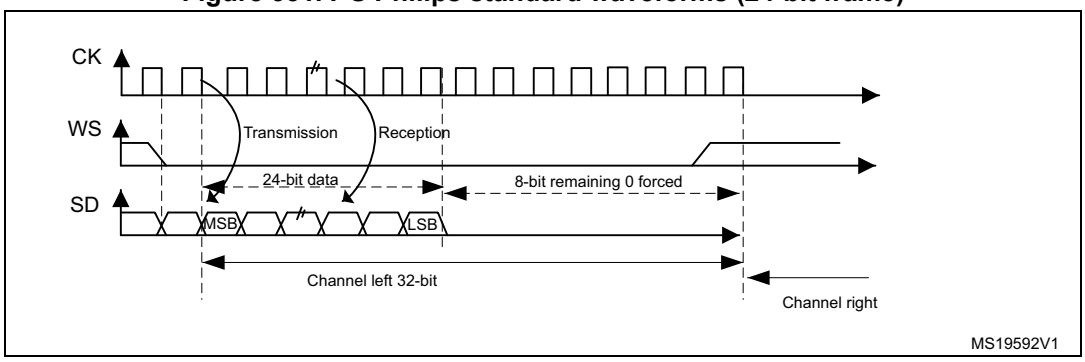
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

Figure 350. I^{2}S Philips protocol waveforms (16/32-bit full accuracy)



Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

Figure 351. I^{2}S Philips standard waveforms (24-bit frame)

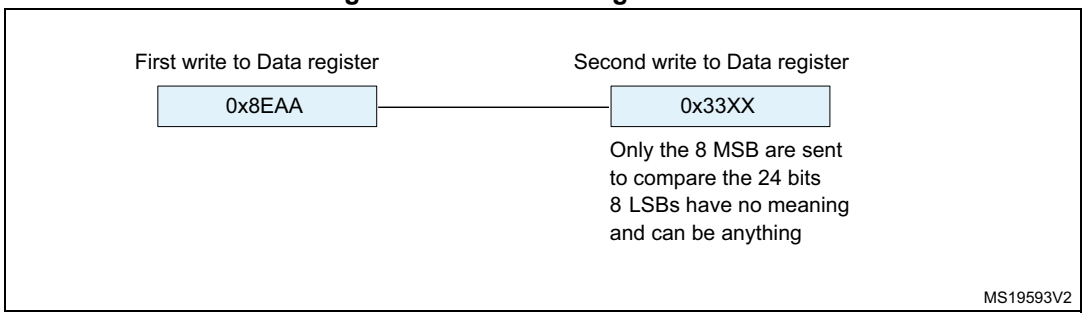


This mode needs two write or read operations to/from the SPIx_DR register.

- In transmission mode:

  If 0x8EAA33 has to be sent (24-bit):

Figure 352. Transmitting 0x8EAA33



- In reception mode:
  If data 0x8EAA33 is received:

Figure 353. Receiving 0x8EAA33

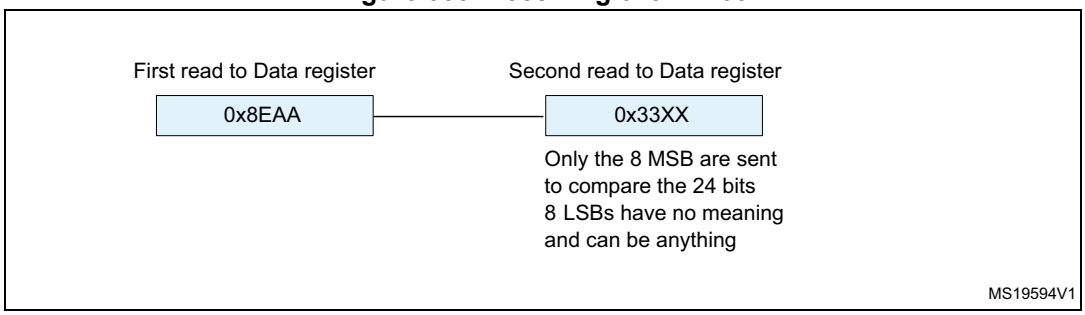
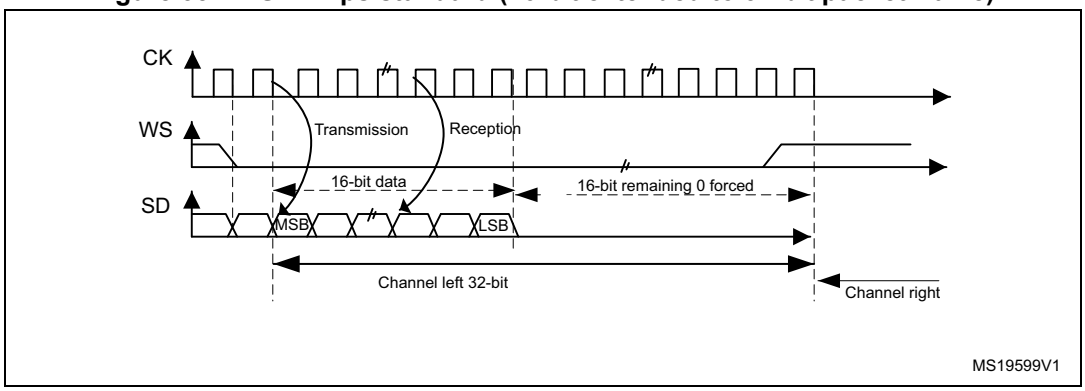


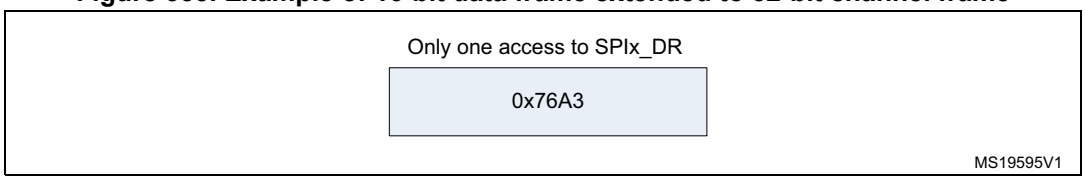
Figure 354. I²S Philips standard (16-bit extended to 32-bit packet frame)



When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in *Figure 355* is required.

Figure 355. Example of 16-bit data frame extended to 32-bit channel frame



For transmission, each time an MSB is written to SPIx_DR, the TXE flag is set and its interrupt, if allowed, is generated to load the SPIx_DR register with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

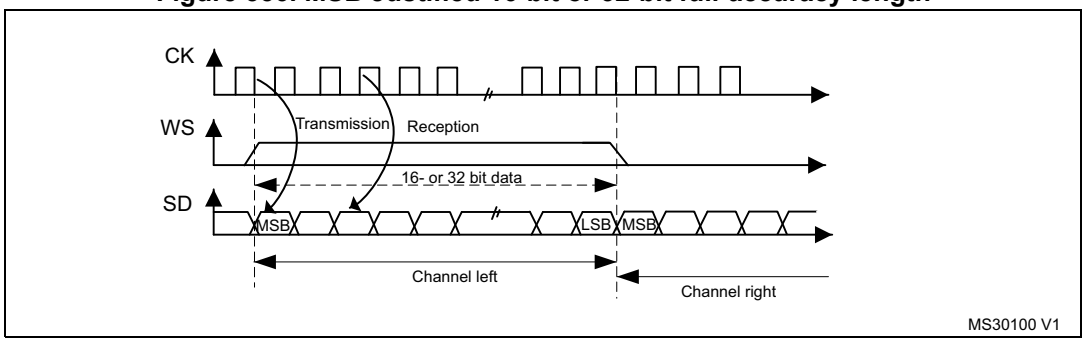
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

MSB justified standard

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

Figure 356. MSB Justified 16-bit or 32-bit full-accuracy length



Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

Figure 357. MSB justified 24-bit frame length

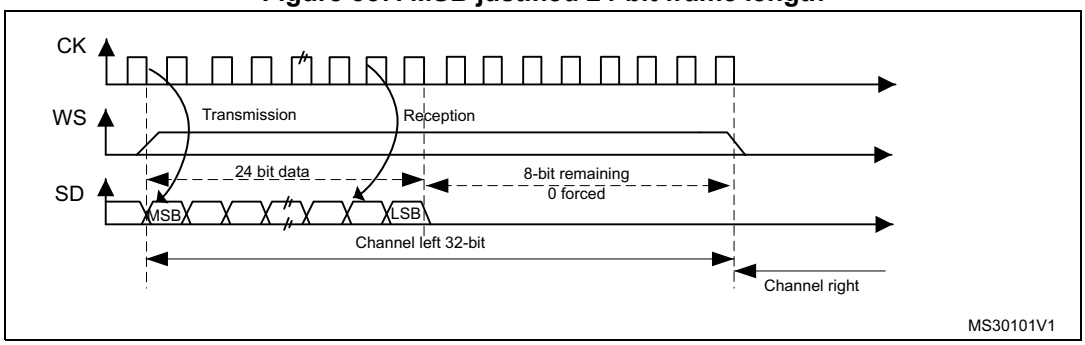
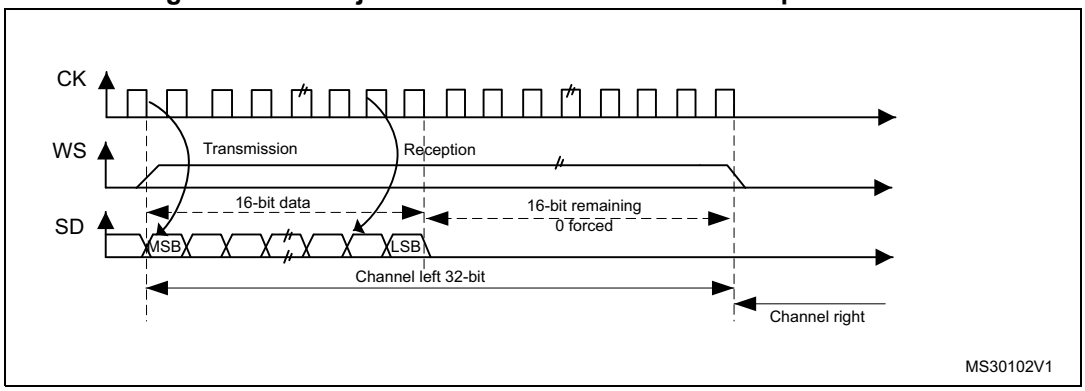


Figure 358. MSB justified 16-bit extended to 32-bit packet frame



LSB justified standard

This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

The sampling of the input and output signals is the same as for the I^2S Philips standard.

Figure 359. LSB justified 16-bit or 32-bit full-accuracy

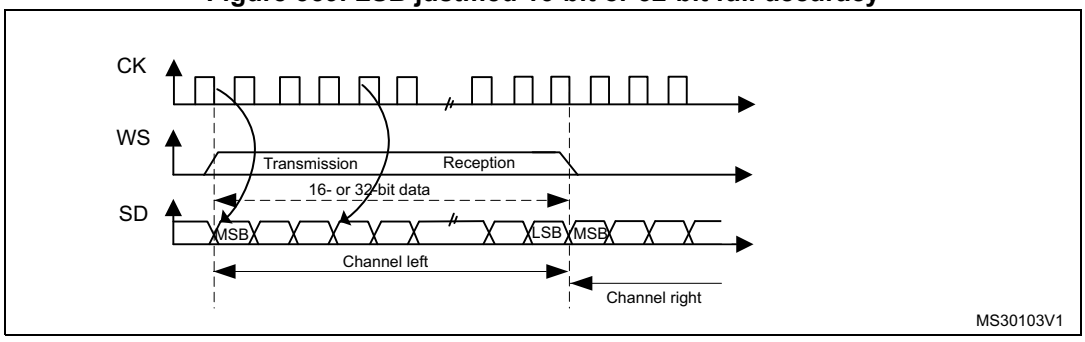
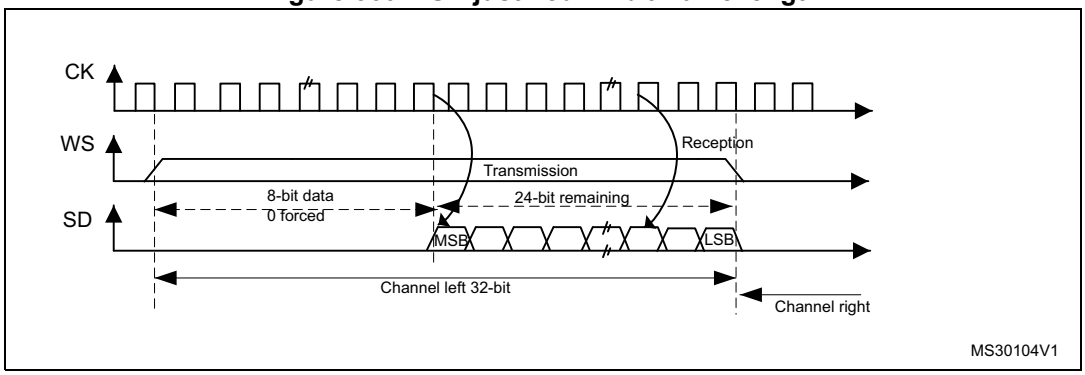


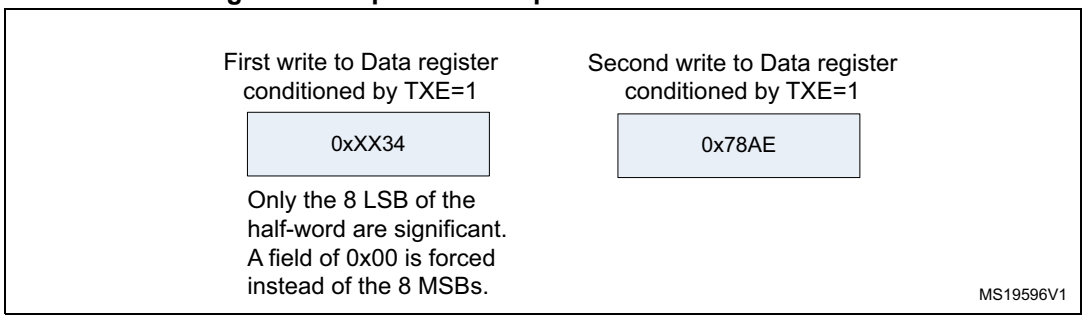
Figure 360. LSB justified 24-bit frame length



- In transmission mode:

  If data 0x3478AE have to be transmitted, two write operations to the SPIx_DR register are required by software or by DMA. The operations are shown below.

Figure 361. Operations required to transmit 0x3478AE



- In reception mode:

  If data 0x3478AE are received, two successive read operations from the SPIx_DR register are required on each RXNE event.

Figure 362. Operations required to receive 0x3478AE

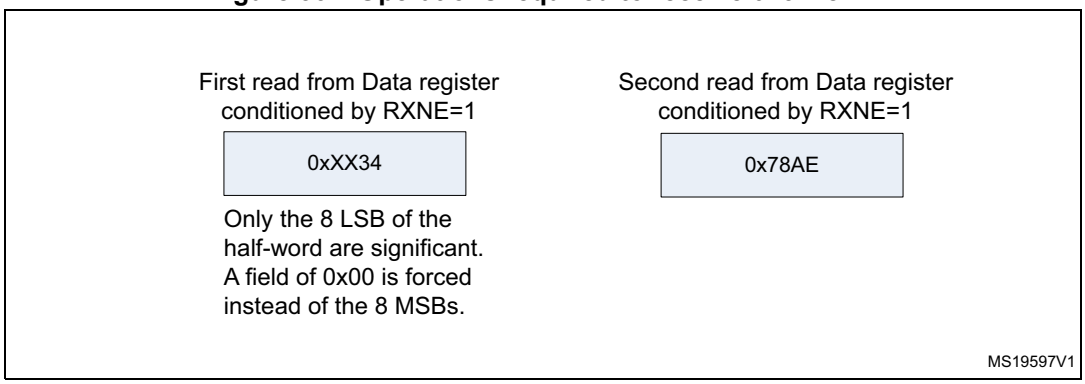
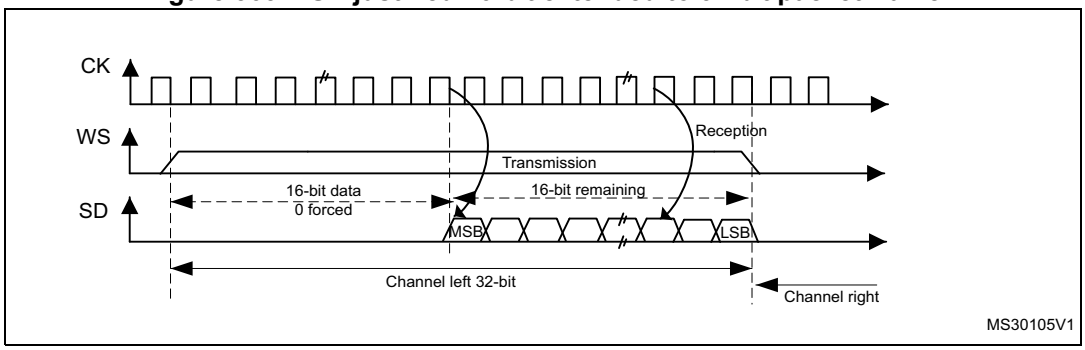


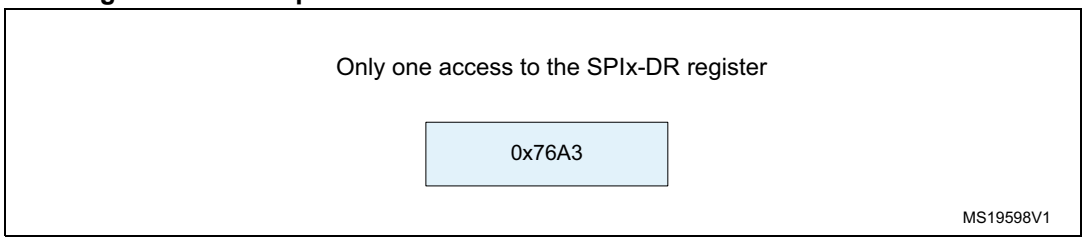
Figure 363. LSB justified 16-bit extended to 32-bit packet frame



When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, Only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in *Figure 364* is required.

Figure 364. Example of 16-bit data frame extended to 32-bit channel frame



In transmission mode, when a TXE event occurs, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). The TXE flag is set again as soon as the effective data (0x76A3) is sent on SD.

In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

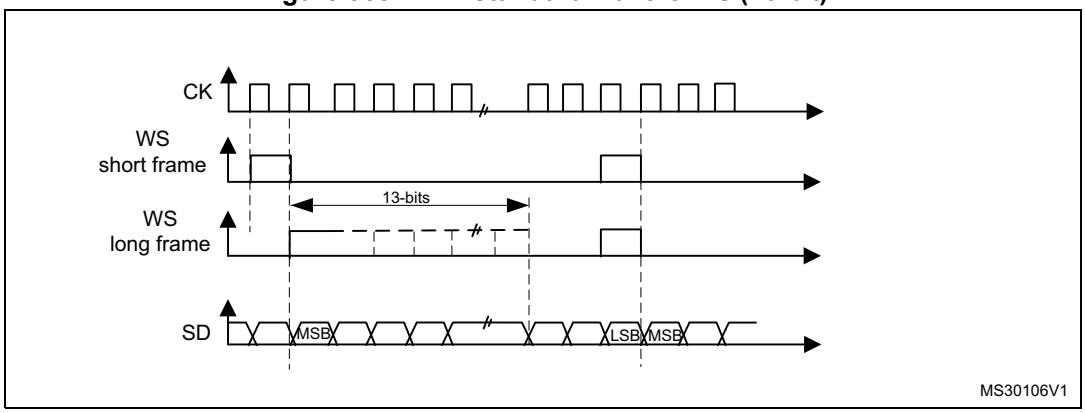
PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPIx_I2SCFGR register.

In PCM mode, the output signals (WS, SD) are sampled on the rising edge of CK signal. The input signals (WS, SD) are captured on the falling edge of CK.

Note that CK and WS are configured as output in MASTER mode.

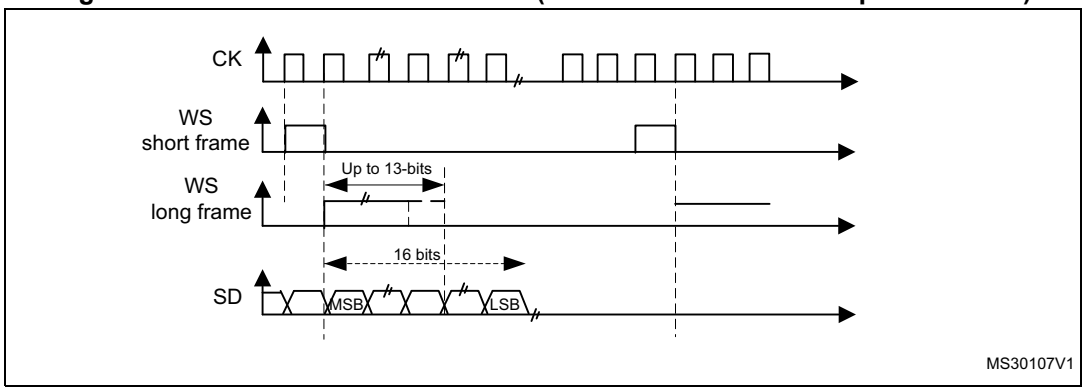
Figure 365. PCM standard waveforms (16-bit)



For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

For short frame synchronization, the WS synchronization signal is only one cycle long.

Figure 366. PCM standard waveforms (16-bit extended to 32-bit packet frame)

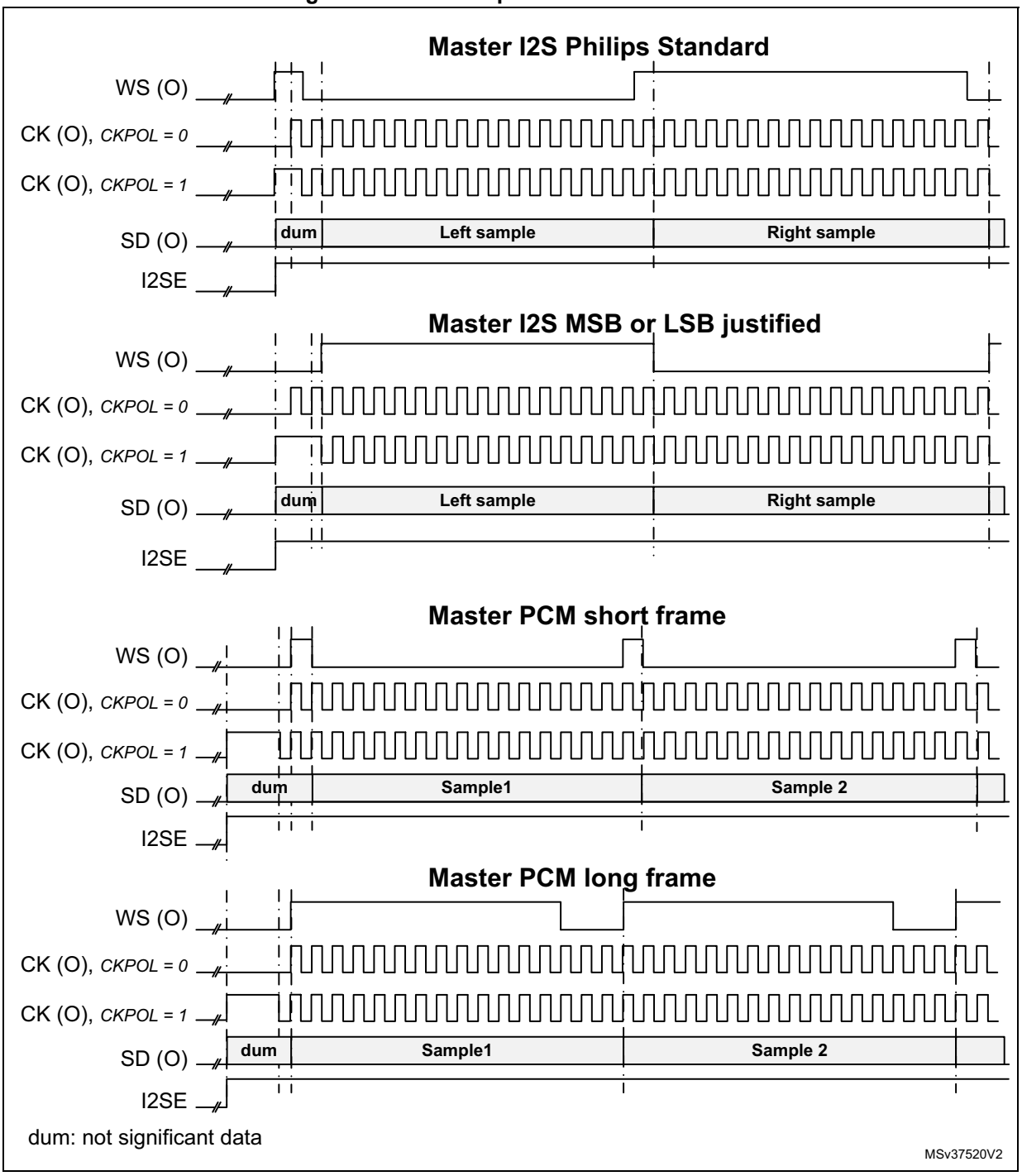


Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPIx_I2SCFGR register) even in slave mode.

35.7.3 Start-up description

The*Figure 367* shows how the serial interface is handled in MASTER mode, when the SPI/I2S is enabled (via I2SE bit). It shows as well the effect of CKPOL on the generated signals.

Figure 367. Start sequence in master mode



In slave mode, the way the frame synchronization is detected, depends on the value of ASTRTEN bit.

If ASTRTEN = 0, when the audio interface is enabled (I2SE = 1), then the hardware waits for the appropriate transition on the incoming WS signal, using the CK signal.

The appropriate transition is a falling edge on WS signal when I²S Philips Standard is used, or a rising edge for other standards. The falling edge is detected by sampling first WS to 1 and then to 0, and vice-versa for the rising edge detection.

If ASTRTEN = 1, the user has to enable the audio interface before the WS becomes active. This means that the I2SE bit must be set to 1 when WS = 1 for I²S Philips standard, or when WS = 0 for other standards.

35.7.4 Clock generator

The I²S bit rate determines the data flow on the I²S data line and the I²S clock signal frequency.

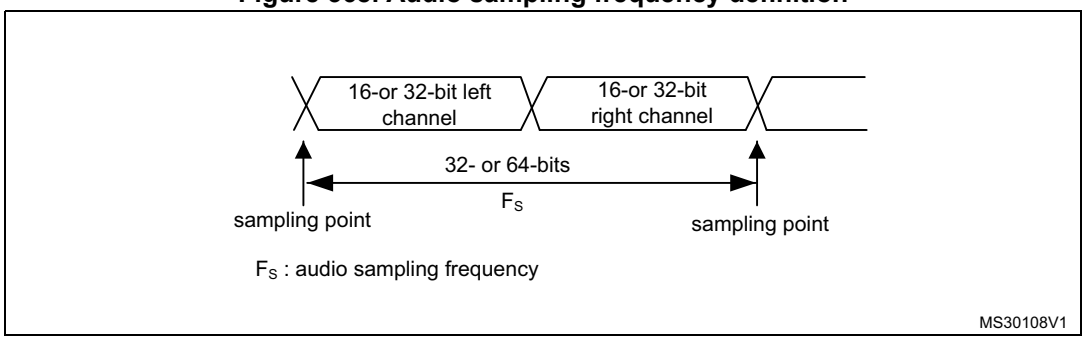
I²S bit rate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I²S bit rate is calculated as follows:

$I^2S \text{ bit rate} = 16 \times 2 \times f_S$

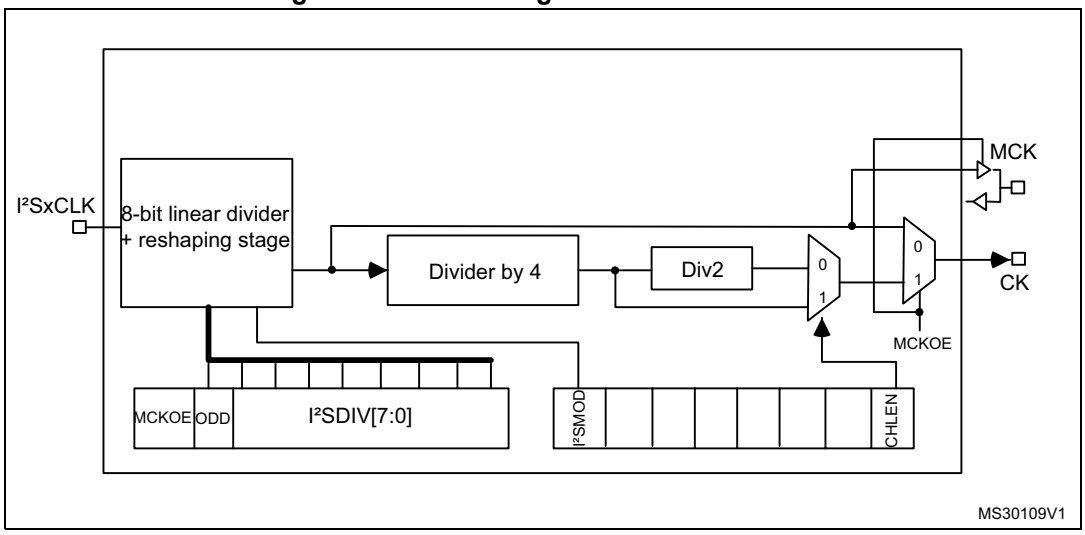
It is: $I^2S \text{ bit rate} = 32 \times 2 \times f_S$ if the packet length is 32-bit wide.

Figure 368. Audio sampling frequency definition



When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency.

Figure 369. I²S clock generator architecture



1. Where x can be 2 or 3.

Figure 369 presents the communication clock architecture. The I2SxCLK clock is provided by the reset and clock controller (RCC) of the product. The I2SxCLK clock can be asynchronous with respect to the SPI/I2S APB clock.

Warning: In addition, it is mandatory to keep the I2SxCLK frequency higher or equal to the APB clock used by the SPI/I2S block. If this condition is not respected the SPI/I2S does not work properly.

The audio sampling frequency may be 192 kHz, 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range).

In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

For I²S modes:

When the master clock is generated (MCKOE in the SPIx_I2SPR register is set):

$$Fs = \frac{F_{I2SxCLK}}{256 \times ((2 \times I2SDIV) + ODD)}$$

When the master clock is disabled (MCKOE bit cleared):

$$Fs = \frac{F_{I2SxCLK}}{32 \times (CHLEN + 1) \times ((2 \times I2SDIV) + ODD)}$$

CHLEN = 0 when the channel frame is 16-bit wide and,

CHLEN = 1 when the channel frame is 32-bit wide.

For PCM modes:

When the master clock is generated (MCKOE in the SPIx_I2SPR register is set):

$$Fs = \frac{F_{I2SxCLK}}{128 \times ((2 \times I2SDIV) + ODD)}$$

When the master clock is disabled (MCKOE bit cleared):

$$Fs = \frac{F_{I2SxCLK}}{16 \times (CHLEN + 1) \times ((2 \times I2SDIV) + ODD)}$$

CHLEN = 0 when the channel frame is 16-bit wide and,

CHLEN = 1 when the channel frame is 32-bit wide.

Where F_S is the audio sampling frequency, and $F_{I2SxCLK}$ is the frequency of the kernel clock provided to the SPI/I2S block.

Note: I2SDIV must be strictly higher than 1.

The following table provides example precision values for different clock configurations.

Note: Other configurations are possible that allow optimum clock precision.

Table 245. Audio-frequency precision using 48 MHz clock derived from HSE(1)

SYSCLK (MHz)	Data length	I2SDIV	I2SODD	MCLK	Target fs (Hz)	Real fs (kHz)	Error
48	16	8	0	No	96000	93750	2.3438%
48	32	4	0	No	96000	93750	2.3438%
48	16	15	1	No	48000	48387.0968	0.8065%
48	32	8	0	No	48000	46875	2.3438%
48	16	17	0	No	44100	44117.647	0.0400%
48	32	8	1	No	44100	44117.647	0.0400%
48	16	23	1	No	32000	31914.8936	0.2660%
48	32	11	1	No	32000	32608.696	1.9022%
48	16	34	0	No	22050	22058.8235	0.0400%
48	32	17	0	No	22050	22058.8235	0.0400%
48	16	47	0	No	16000	15957.4468	0.2660%
48	32	23	1	No	16000	15957.447	0.2660%
48	16	68	0	No	11025	11029.4118	0.0400%
48	32	34	0	No	11025	11029.412	0.0400%
48	16	94	0	No	8000	7978.7234	0.2660%
48	32	47	0	No	8000	7978.7234	0.2660%
48	16	2	0	Yes	48000	46875	2.3430%
48	32	2	0	Yes	48000	46875	2.3430%
48	16	2	0	Yes	44100	46875	6.2925%
48	32	2	0	Yes	44100	46875	6.2925%
48	16	3	0	Yes	32000	31250	2.3438%
48	32	3	0	Yes	32000	31250	2.3438%
48	16	4	1	Yes	22050	20833.333	5.5178%
48	32	4	1	Yes	22050	20833.333	5.5178%
48	16	6	0	Yes	16000	15625	2.3438%
48	32	6	0	Yes	16000	15625	2.3438%
48	16	8	1	Yes	11025	11029.4118	0.0400%
48	32	8	1	Yes	11025	11029.4118	0.0400%
48	16	11	1	Yes	8000	8152.17391	1.9022%
48	32	11	1	Yes	8000	8152.17391	1.9022%

1. This table gives only example values for different clock configurations. Other configurations allowing optimum clock precision are possible.

35.7.5 I²S master mode

The I2S can be configured in master mode. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, controlled by the MCKOE bit in the SPIx_I2SPR register.

Procedure

1. Select the I2SDIV[7:0] bits in the SPIx_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPIx_I2SPR register also has to be defined.
2. Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPIx_I2SPR register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to *Section 35.7.4: Clock generator*).
3. Set the I2SMOD bit in the SPIx_I2SCFGR register to activate the I2S functions and choose the I²S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the CHLEN bit. Select also the I²S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.
4. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
5. The I2SE bit in SPIx_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPIx_I2SPR is set.

Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Lets assume the first data written into the Tx buffer corresponds to the left channel data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the right channel have to be written into the Tx buffer. The CHSIDE flag indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high.

A full frame has to be considered as a left channel data transmission followed by a right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

For more details about the write operations depending on the I²S standard mode selected, refer to *Section 35.7.2: Supported audio protocols*).

To ensure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission.

To switch off the I2S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for transmission mode except for the point 3 (refer to the procedure described in *Section 35.7.5: I²S master mode*), where the configuration should set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPIx_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I2S cell.

For more details about the read operations depending on the I²S standard mode selected, refer to *Section 35.7.2: Supported audio protocols*.

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S, specific actions are required to ensure that the I2S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB justified mode (I2SSTD = 10)
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait 17 I2S clock cycles (using a software loop)
  c) Disable the I2S (I2SE = 0)
- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, I²S or PCM modes (I2SSTD = 00, I2SSTD = 01 or I2SSTD = 11, respectively)
  a) Wait for the last RXNE
  b) Then wait 1 I2S clock cycle (using a software loop)
  c) Disable the I2S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I2S:
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait one I2S clock cycle (using a software loop)
  c) Disable the I2S (I2SE = 0)

Note: The BSY flag is kept low during transfers.

35.7.6 I²S slave mode

For the slave configuration, the I2S can be configured in transmission or reception mode. The operating mode is following mainly the same rules as described for the I²S master

configuration. In slave mode, there is no clock to be generated by the I2S interface. The clock and WS signals are input from the external master connected to the I2S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPIx_I2SCFGR register to select I^2S mode and choose the I^2S standard through the I2SSTD[1:0] bits, the data length through the DATLEN[1:0] bits and the number of bits per channel for the frame configuring the CHLEN bit. Select also the mode (transmission or reception) for the slave through the I2SCFG[1:0] bits in SPIx_I2SCFGR register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
3. The I2SE bit in SPIx_I2SCFGR register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I2S data register has to be loaded before the master initiates the communication.

For the I2S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I2S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

Note: The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

Note that the TXE flag should be checked to be at 1 before attempting to write the Tx buffer.

For more details about the write operations depending on the I^2S standard mode selected, refer to *Section 35.7.2: Supported audio protocols*.

To secure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPIx_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPIx_CR2 register, an interrupt is generated when the UDR flag in the SPIx_SR register goes high. In this case, it is mandatory to switch off the I2S and to restart a data transfer starting from the left channel.

To switch off the I2S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in *Section 35.7.6: I²S slave mode*), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.

Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPIx_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPIx_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer.

The CHSIDE flag is updated each time data are received to be read from the SPIx_DR register. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

For more details about the read operations depending the I²S standard mode selected, refer to *Section 35.7.2: Supported audio protocols*.

If data are received while the preceding received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

Note: The external master components should have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

35.7.7 I2S status flags

Three status flags are provided for the application to fully monitor the state of the I2S bus.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). It indicates the state of the communication layer of the I2S.

When BSY is set, it indicates that the I2S is busy communicating. There is one exception in master receive mode (I2SCFG = 11) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software needs to disable the I2S. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is set when a transfer starts, except when the I2S is in master receiver mode.

The BSY flag is cleared:

- When a transfer completes (except in master transmit mode, in which the communication is supposed to be continuous)
- When the I2S is disabled

When communication is continuous:

- In master transmit mode, the BSY flag is kept high during all the transfers
- In slave mode, the BSY flag goes low for one I2S clock cycle between each transfer

Note: Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

Tx buffer empty flag (TXE)

When set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can then be loaded into it. The TXE flag is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I2S is disabled (I2SE bit is reset).

RX buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when SPIx_DR register is read.

Channel Side flag (CHSIDE)

In transmission mode, this flag is refreshed when TXE goes high. It indicates the channel side to which the data to transfer on SD has to belong. In case of an underrun error event in slave transmission mode, this flag is not reliable and I2S needs to be switched off and switched on before resuming the communication.

In reception mode, this flag is refreshed when data are received into SPIx_DR. It indicates from which channel side data have been received. Note that in case of error (like OVR) this flag becomes meaningless and the I2S should be reset by disabling and then enabling it (with configuration if it needs changing).

This flag has no meaning in the PCM standard (for both Short and Long frame modes).

When the OVR or UDR flag in the SPIx_SR is set and the ERRIE bit in SPIx_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPIx_SR status register (once the interrupt source has been cleared).

35.7.8 I2S error flags

There are three error flags for the I2S cell.

Underrun flag (UDR)

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPIx_DR. It is available when the I2SMOD bit in the SPIx_I2SCFGR register is set. An interrupt may be generated if the ERRIE bit in the SPIx_CR2 register is set.
The UDR bit is cleared by a read operation on the SPIx_SR register.

Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from the SPIx_DR register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx_CR2 register.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPIx_DR register followed by a read access to the SPIx_SR register.

Frame error flag (FRE)

This flag can be set by hardware only if the I2S is configured in Slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the

synchronization is lost, the following steps are required to recover from this state and resynchronize the external master device with the I2S slave device:

1. Disable the I2S.
2. Enable it again when the correct level is detected on the WS line (WS line is high in I^2S mode or low for MSB- or LSB-justified or PCM modes.

Desynchronization between master and slave devices may be due to noisy environment on the CK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by software when the status register is read.

35.7.9 DMA features

In I^2S mode, the DMA works in exactly the same way as it does in SPI mode. There is no difference except that the CRC feature is not available in I^2S mode since there is no data transfer protection system.

35.8 I2S interrupts

Table 246 provides the list of I2S interrupts.

Table 246. I2S interrupt requests

Interrupt event	Event flag	Enable control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Overrun error	OVR	ERRIE
Underrun error	UDR	
Frame error flag	FRE	

35.9 SPI and I2S registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI_DR in addition can be accessed by 8-bit access.

35.9.1 SPI control register 1 (SPIx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDIOE	CRC EN	CRCN EXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE:** Bidirectional data mode enable.

This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

Note: This bit is not used in I^2S mode.

Bit 14 **BIDIOE:** Output enable in bidirectional mode

This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode.

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used. This bit is not used in I^2S mode.

Bit 13 **CRCEN:** Hardware CRC calculation enable

0: CRC calculation disabled

1: CRC calculation enabled

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation. This bit is not used in I^2S mode.

Bit 12 **CRCNEXT:** Transmit CRC next

0: Next transmit value is from Tx buffer.

1: Next transmit value is from Tx CRC register.

Note: This bit has to be written as soon as the last data is written in the SPIx_DR register. This bit is not used in I^2S mode.

Bit 11 **CRCL:** CRC length

This bit is set and cleared by software to select the CRC length.

0: 8-bit CRC length

1: 16-bit CRC length

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation. This bit is not used in I^2S mode.

Bit 10 **RXONLY:** Receive only mode enabled.

This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receive only mode is active.This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

0: Full-duplex (Transmit and receive)

1: Output disabled (Receive-only mode)

Note: This bit is not used in I^2S mode.

Bit 9 **SSM:** Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

0: Software slave management disabled

1: Software slave management enabled

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 8 **SSI:** Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored.

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 7 **LSBFIRST***:* Frame format

0: data is transmitted / received with the MSB first

1: data is transmitted / received with the LSB first

Note: 1. This bit should not be changed when communication is ongoing.

2. This bit is not used in I^2S mode and SPI TI mode.

Bit 6 **SPE:** SPI enable

0: Peripheral disabled

1: Peripheral enabled

Note: When disabling the SPI, follow the procedure described in Procedure for disabling the SPI on page 1167.

This bit is not used in I^2S mode.

Bits 5:3 **BR[2:0]:** Baud rate control

000: $f_{PCLK}/2$

001: $f_{PCLK}/4$

010: $f_{PCLK}/8$

011: $f_{PCLK}/16$

100: $f_{PCLK}/32$

101: $f_{PCLK}/64$

110: $f_{PCLK}/128$

111: $f_{PCLK}/256$

Note: These bits should not be changed when communication is ongoing.

These bits are not used in I^2S mode.

Bit 2 **MSTR:** Master selection

0: Slave configuration

1: Master configuration

Note: This bit should not be changed when communication is ongoing.

This bit is not used in I^2S mode.

Bit 1 **CPOL:** Clock polarity
0: CK to 0 when idle
1: CK to 1 when idle
Note: This bit should not be changed when communication is ongoing.
This bit is not used in I^2S mode and SPI TI mode except the case when CRC is applied at TI mode.

Bit 0 **CPHA:** Clock phase
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge
Note: This bit should not be changed when communication is ongoing.
This bit is not used in I^2S mode and SPI TI mode except the case when CRC is applied at TI mode.

35.9.2 SPI control register 2 (SPIx_CR2)

Address offset: 0x04

Reset value: 0x0700

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **LDMA_TX:** Last DMA transfer for transmission
This bit is used in data packing mode, to define if the total number of data to transmit by DMA is odd or even. It has significance only if the TXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).
0: Number of data to transfer is even
1: Number of data to transfer is odd
Note: Refer to Procedure for disabling the SPI on page 1167 if the CRCEN bit is set.
This bit is not used in I²S mode.

Bit 13 **LDMA_RX**: Last DMA transfer for reception
This bit is used in data packing mode, to define if the total number of data to receive by DMA is odd or even. It has significance only if the RXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).
0: Number of data to transfer is even
1: Number of data to transfer is odd
Note: Refer to Procedure for disabling the SPI on page 1167 if the CRCEN bit is set.
This bit is not used in I²S mode.

Bit 12 **FRXTH**: FIFO reception threshold
This bit is used to set the threshold of the RXFIFO that triggers an RXNE event
0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit)
1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)
Note: This bit is not used in I^2S mode.

Bits 11:8 **DS[3:0]**: Data size

These bits configure the data length for SPI transfers.

0000: Not used
0001: Not used
0010: Not used
0011: 4-bit
0100: 5-bit
0101: 6-bit
0110: 7-bit
0111: 8-bit
1000: 9-bit
1001: 10-bit
1010: 11-bit
1011: 12-bit
1100: 13-bit
1101: 14-bit
1110: 15-bit
1111: 16-bit

If software attempts to write one of the “Not used” values, they are forced to the value “0111” (8-bit)

Note: These bits are not used in I^2S mode.

Bit 7 **TXEIE:** Tx buffer empty interrupt enable

0: TXE interrupt masked
1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE:** RX buffer not empty interrupt enable

0: RXNE interrupt masked
1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE:** Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode and UDR, OVR, and FRE in I^2S mode).

0: Error interrupt is masked
1: Error interrupt is enabled

Bit 4 **FRF**: Frame format

0: SPI Motorola mode
1 SPI TI mode

Note: This bit must be written only when the SPI is disabled (SPE=0).
This bit is not used in I^2S mode.

Bit 3 **NSSP**: NSS pulse management

This bit is used in master mode only. it allows the SPI to generate an NSS pulse between two consecutive data when doing continuous transfers. In the case of a single data transfer, it forces the NSS pin high level after the transfer.

It has no meaning if CPHA = ’1’, or FRF = ’1’.

0: No NSS pulse
1: NSS pulse generated

Note: 1. This bit must be written only when the SPI is disabled (SPE=0).
2. This bit is not used in I^2S mode and SPI TI mode.

Bit 2 **SSOE:** SS output enable

0: SS output is disabled in master mode and the SPI interface can work in multimaster configuration

1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multimaster environment.

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 1 **TXDMAEN:** Tx buffer DMA enable

When this bit is set, a DMA request is generated whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

Bit 0 **RXDMAEN:** Rx buffer DMA enable

When this bit is set, a DMA request is generated whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

35.9.3 SPI status register (SPIx_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
			r	r	r	r	r	r	r	r	rc_w0	r	r	r	r

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:11 **FTLVL[1:0]:** FIFO transmission level

These bits are set and cleared by hardware.

00: FIFO empty

01: 1/4 FIFO

10: 1/2 FIFO

11: FIFO full (considered as FULL when the FIFO threshold is greater than 1/2)

Note: This bit is not used in I^2S mode.

Bits 10:9 **FRLVL[1:0]**: FIFO reception level

These bits are set and cleared by hardware.

00: FIFO empty

01: 1/4 FIFO

10: 1/2 FIFO

11: FIFO full

Note: These bits are not used in I²S mode and in SPI receive-only mode while CRC calculation is enabled.

Bit 8 **FRE**: Frame format error

This flag is used for SPI in TI slave mode and I^2S slave mode. Refer to *Section 35.5.11: SPI error flags* and *Section 35.7.8: I2S error flags*.

This flag is set by hardware and reset when SPIx_SR is read by software.

0: No frame format error

1: A frame format error occurred

Bit 7 **BSY:** Busy flag
0: SPI (or I2S) not busy
1: SPI (or I2S) is busy in communication or Tx buffer is not empty
This flag is set and cleared by hardware.
Note: The BSY flag must be used with caution: refer to Section 35.5.10: SPI status flags and Procedure for disabling the SPI on page 1167.

Bit 6 **OVR:** Overrun flag
0: No overrun occurred
1: Overrun occurred
This flag is set by hardware and reset by a software sequence. Refer to *I2S error flags on page 1199* for the software sequence.

Bit 5 **MODF:** Mode fault
0: No mode fault occurred
1: Mode fault occurred
This flag is set by hardware and reset by a software sequence. Refer to *Section : Mode fault (MODF) on page 1177* for the software sequence.
Note: This bit is not used in I^2S mode.

Bit 4 **CRCERR:** CRC error flag
0: CRC value received matches the SPIx_RXCRCR value
1: CRC value received does not match the SPIx_RXCRCR value
Note: This flag is set by hardware and cleared by software writing 0.
This bit is not used in I^2S mode.

Bit 3 **UDR:** Underrun flag
0: No underrun occurred
1: Underrun occurred
This flag is set by hardware and reset by a software sequence. Refer to *I2S error flags on page 1199* for the software sequence.
Note: This bit is not used in SPI mode.

Bit 2 **CHSIDE**: Channel side
0: Channel Left has to be transmitted or has been received
1: Channel Right has to be transmitted or has been received
Note: This bit is not used in SPI mode. It has no significance in PCM mode.

Bit 1 **TXE:** Transmit buffer empty
0: Tx buffer not empty
1: Tx buffer empty

Bit 0 **RXNE:** Receive buffer not empty
0: Rx buffer empty
1: Rx buffer not empty

35.9.4 SPI data register (SPIx_DR)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DR[15:0]:** Data register

Data received or to be transmitted

The data register serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses TxFIFO (See *Section 35.5.9: Data transmission and reception procedures*).

Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.

35.9.5 SPI CRC polynomial register (SPIx_CRCPR)

Address offset: 0x10

Reset value: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CRCPOLY[15:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0x0007) is the reset value of this register. Another polynomial can be configured as required.

Note: The polynomial value should be odd only. No even value is supported.

35.9.6 SPI Rx CRC register (SPIx_RXCRCR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **RXCRC[15:0]:** Rx CRC register

When CRC calculation is enabled, the RXCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPIx_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY Flag is set could return an incorrect value. These bits are not used in I^2S mode.

35.9.7 SPI Tx CRC register (SPIx_TXCRCR)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **TXCRC[15:0]:** Tx CRC register

When CRC calculation is enabled, the TXCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPIx_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY flag is set could return an incorrect value. These bits are not used in I^2S mode.

35.9.8 SPIx_I2S configuration register (SPIx_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ASTRTEN	I2SMOD	I2SE	I2SCFG[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		CKPOL	DATLEN[1:0]		CHLEN
			rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ASTRTEN**: Asynchronous start enable.

0: The Asynchronous start is disabled.

When the I2S is enabled in slave mode, the hardware starts the transfer when the I2S clock is received and an appropriate transition is detected on the WS signal.

1: The Asynchronous start is enabled.

When the I2S is enabled in slave mode, the hardware starts the transfer when the I2S clock is received and the appropriate level is detected on the WS signal.

*Note: The appropriate **transition** is a falling edge on WS signal when I^2S Philips Standard is used, or a rising edge for other standards.*

*The appropriate **level** is a low level on WS signal when I^2S Philips Standard is used, or a high level for other standards.*

Please refer to Section 35.7.3: Start-up description for additional information.

Bit 11 **I2SMOD**: I2S mode selection

0: SPI mode is selected

1: I2S mode is selected

Note: This bit should be configured when the SPI is disabled.

Bit 10 **I2SE**: I2S enable

0: I2S peripheral is disabled

1: I2S peripheral is enabled

Note: This bit is not used in SPI mode.

Bits 9:8 **I2SCFG[1:0]**: I2S configuration mode

00: Slave - transmit

01: Slave - receive

10: Master - transmit

11: Master - receive

Note: These bits should be configured when the I2S is disabled.

They are not used in SPI mode.

Bit 7 **PCMSYNC**: PCM frame synchronization

0: Short frame synchronization

1: Long frame synchronization

Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used).

It is not used in SPI mode.

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I2S standard selection

00: I^2S Philips standard

01: MSB justified standard (left justified)

10: LSB justified standard (right justified)

11: PCM standard

For more details on I^2S standards, refer to Section 35.7.2 on page 1183

Note: For correct operation, these bits should be configured when the I2S is disabled.

They are not used in SPI mode.

Bit 3 **CKPOL**: Inactive state clock polarity
0: I2S clock inactive state is low level
1: I2S clock inactive state is high level
Note: For correct operation, this bit should be configured when the I2S is disabled.
It is not used in SPI mode.
The bit CKPOL does not affect the CK edge sensitivity used to receive or transmit the SD and WS signals.

Bits 2:1 **DATLEN[1:0]**: Data length to be transferred
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: For correct operation, these bits should be configured when the I2S is disabled.
They are not used in SPI mode.

Bit 0 **CHLEN**: Channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide
The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in.
Note: For correct operation, this bit should be configured when the I2S is disabled.
It is not used in SPI mode.

35.9.9 SPIx_I2S prescaler register (SPIx_I2SPR)

Address offset: 0x20

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **MCKOE**: Master clock output enable
0: Master clock output is disabled
1: Master clock output is enabled
Note: This bit should be configured when the I2S is disabled. It is used only when the I2S is in master mode.
It is not used in SPI mode.

Bit 8 **ODD**: Odd factor for the prescaler
0: Real divider value is = I2SDIV *2
1: Real divider value is = (I2SDIV * 2) + 1
Refer to *Section 35.7.3 on page 1190*.
Note: This bit should be configured when the I2S is disabled. It is used only when the I2S is in master mode.
It is not used in SPI mode.

Bits 7:0 **I2SDIV[7:0]**: I2S linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to *Section 35.7.3 on page 1190*.

Note: These bits should be configured when the I2S is disabled. They are used only when the I2S is in master mode.

They are not used in SPI mode.

35.9.10 SPI/I2S register map

Table 247 shows the SPI/I2S register map and reset values.

Table 247. SPI/I2S register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPIx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPIx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	*SSOE*	TXDMAEN	RXDMAEN
	Reset value																		0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0x08	SPIx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	1	0
0x0C	SPIx_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPIx_CRCPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	SPIx_RXCRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPIx_TXCRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPIx_I2SCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASTRTEN	I2SMOD	I2SE	I2SCFG[1:0]		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN[1:0]		CHLEN
	Reset value																				0	0	0	0	0	0		0	0	0	0	0	0
0x20	SPIx_I2SPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
	Reset value																							0	0	0	0	0	0	0	0	1	0

Refer to *Section 2.4 on page 62* for the register boundary addresses.

36 Debug support (DBG)

36.1 DBG introduction and main features

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core in the system
- Code execution tracing
- Software instrumentation

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm CoreSight™ components.

- General features:
  - SWJ-DP: JTAG/Serial-wire debug port
  - AHB-AP: AHB access port
- CPU debug features
  - ROM table (see *Section 36.7*)
  - System control space (SCS)
  - Breakpoint unit (FPB) (see *Section 36.8*)
  - Data watchpoint and trace unit (DWT) (see *Section 36.6*)
  - Instrumentation trace macrocell (ITM) (see *Section 36.9*)
  - Trace port interface unit (TPIU) (see *Section 36.10*)

CPU debug features are accessible by the debugger via the CPU AHB-AP.

Additional information can be found in the Arm® documents referenced in *Section 36.12*.

Device level debug features are controlled in the DBGMCU (see *Section 36.11*).

36.2 DBG use cases

The trace and debug system is designed to support a variety of typical use cases:

- **Low-cost trace**

  Limited trace capability is available over the single-wire debug output. This supports code instrumentation using `printf`, tracing of data and address watchpoints, interrupt detection and program counter sampling. Single-wire trace can be maintained even when the processor is switched off or clock-stopped.
- **Breakpoint debugging** of the core

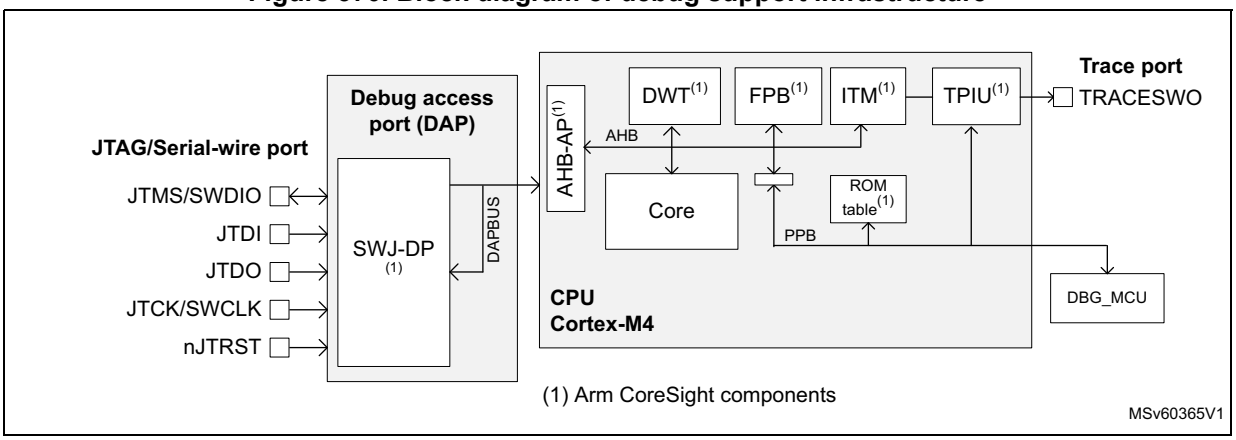
  The processor core can be debugged using equipment connected to the JTAG/SWD debug port. This enables, among others, breakpoint and watchpoint setting, code stepping and memory access.
- **Tracing code execution** via the trace port

  Trace information from the CPU (Cortex-M4) is combined into a single trace stream and sent to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.

36.3 DBG functional description

36.3.1 DBG block diagram

Figure 370. Block diagram of debug support infrastructure



36.3.2 DBG pins and internal signals

Table 248. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		Serial-wire debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	Serial wire data in/out	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial wire clock	PA14
JTDI	I	JTAG test data input	-	-	PA15
JTDO/TRACESWO(1)	O	JTAG test data output	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

1. Debug access port JTDO and Trace port TRACESWO are multiplexed on a single device GPIO pin.

Table 249. Single-wire trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Single wire trace asynchronous data out	PB3(1)

1. TRACESWO is multiplexed with JTDO. This means that single-wire trace is only available when using the Serial-wire debug interface, and not when using JTAG.

36.3.3 DBG reset and clocks

The debug port (SWJ-DP) is reset by a power-on reset or an OBL (option byte loading) reset, and when waking up from Standby mode.

The debugger supplies the clock for the debug port via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both Serial-wire and JTAG modes, as well as to operate the state machines and internal logic of the debug port.

It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DAPCLK domain, which covers the rest of the SWJ-DP.

The DAPCLK is a gated version of the system HCLK3.

The DAPCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the DP_CTRLSTATR register. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in the DP_CTRLSTATR register. DAPCLK is disabled at power up, after OBL, and after a wakeup from Standby. DAPCLK must be disabled when the debugger is disconnected to avoid wasting energy.

The debug components included in the processor (such as ITM, DWG, FPB) are clocked with the core clock.

36.3.4 DBG power domains

The debug components are located in the core power domain. This means that debugger connection is not possible in Shutdown or Standby low-power modes. To avoid losing the connection when the device enters Standby mode, it is possible to maintain the power to the core by setting a bit in the microcontroller debug unit (DBGMCU). This also keeps the processor clocks active and hold off the reset, so that the debug session is maintained.

36.3.5 DBG low-power modes

The STM32WLEx devices include power saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off, or the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power saving mode emulation has been implemented. If emulation is enabled for a domain, the domain still enters power saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power saving mode, but the debugger does not lose the connection.

Emulation mode is programmed in the DBGMCU. For more information refer to *Section 36.11: Microcontroller debug unit (DBGMCU)*.

36.3.6 Serial-wire and JTAG debug port

The Serial-wire and JTAG debug port (SWJ-DP) is a CoreSight component that implements an external access port for connecting debugging equipment.

The two following types of interface can be configured:

- a 5-pin standard JTAG interface (JTAG-DP)
- a 2-pin (clock + data) Serial-wire debug port (SW-DP)

The two modes are mutually exclusive since they share the same I/O pins.

By default the JTAG-DP is selected after a system or a power-on reset. The five I/O pins are configured by hardware in debug alternative function mode.

The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

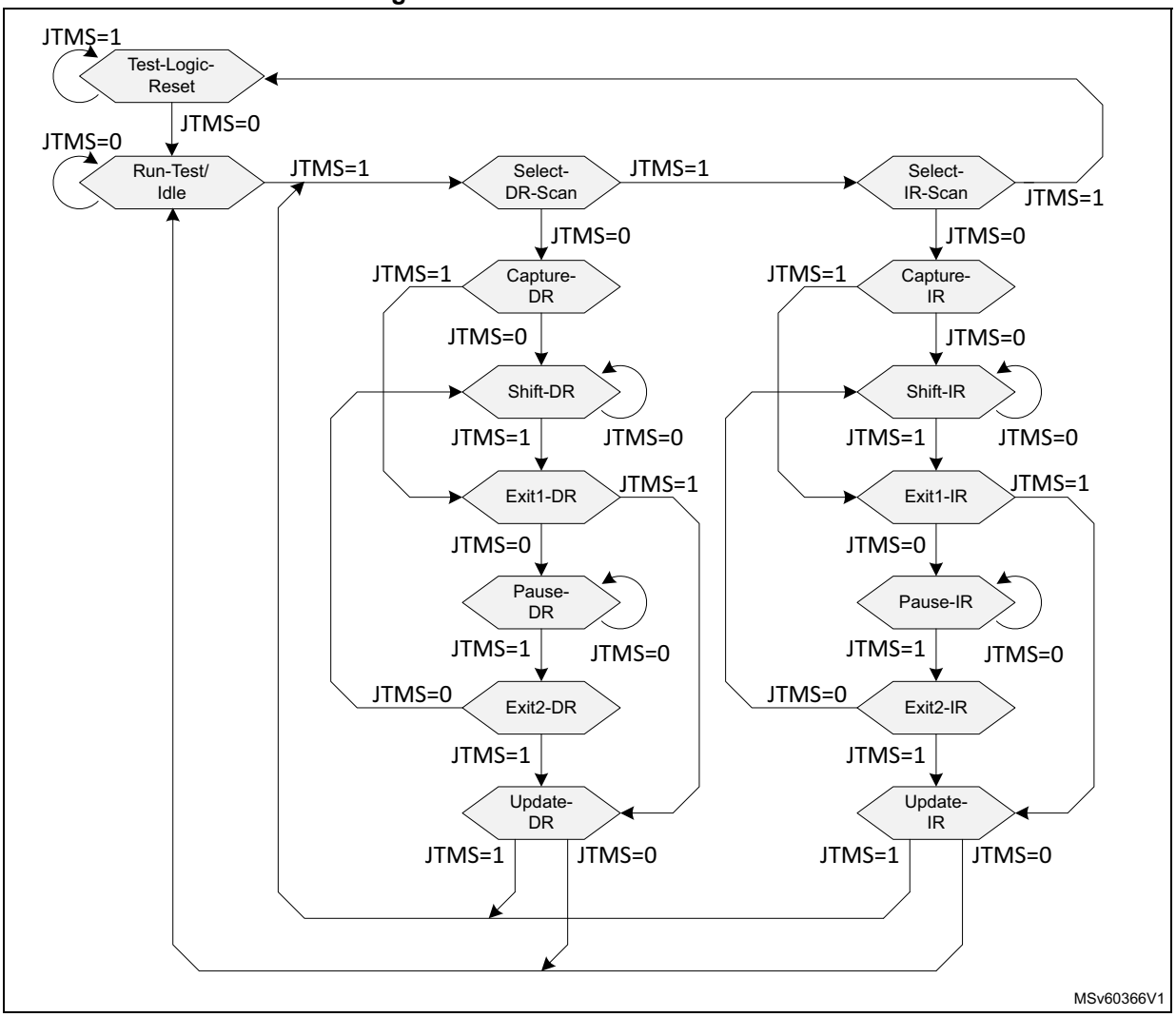
In SW-DP mode, the unused JTAG pins JTDI, JTDO and nJTRST can be used for other functions.

Note: All SWJ port I/Os can be reconfigured to other functions by software but debugging is no longer possible.

36.3.7 JTAG debug port

The JTAG debug port (JTAG-DP) implements a TAP state machine (TAPSM) shown in *Figure 371*, based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR) and the other one with a number of data registers (DR).

Figure 371. JTAG TAP state machine



The operation of the JTAG-DP is as follows:

- When the TAPSM goes through the Capture-IR state, 0b0001 is transferred onto the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
- While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that, on the first tick:
  - The LSB of the IR scan chain is output on JTDO.
  - Bit[n] of the IR scan chain is transferred to bit[n-1].
  - The value on JTDI is transferred to the MSB of the IR scan chain.
- When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the instruction register.
- When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers onto one of the DR scan chains, connected between JTDI and JTDO.
- The value held in the instruction register determines which data register and associated DR scan chain are selected.

- This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shift in the Shift-IR state.
- When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected data register.
- When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in the instruction register.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in *Table 250*.

Table 250. JTAG-DP data registers

IR instruction	Data register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – Bits 34:1 = Reserved – Bit 0 = APABORT: write 1 to generate an AP abort.
1001	(BYPASS)	1	Reserved: BYPASS selected
1010	DPACC	35	Debug port access register Initiates the debug port and gives access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register Bit 0 = RnW = Read request (1) or write request (0) – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge 010 = OK/FAULT 001 = WAIT OTHER = reserved
1011	APACC	35	Access port access register Initiates an access port and gives access to an access port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register Bit 0 = RnW= Read request (1) or write request (0) – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge 010 = OK/FAULT 001 = WAIT OTHER = reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected

Table 250. JTAG-DP data registers (continued)

IR instruction	Data register	Scan chain length	Description
1110	IDCODE	32	ID code 0x6BA0 0477: Arm® JTAG debug port ID code
1111	BYPASS	1	Bypass A single JTCK cycle delay is inserted between JTDI and JTDO.

Data registers are described in more detail in the Arm® Debug Interface Architecture Specification *[1]*.

36.3.8 Serial-wire debug port

The Serial-wire debug (SWD) protocol uses the two following pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock.

Each transfer comprises the three phases listed below:

1. a packet request (8 bits) transmitted by the host (see *Table 251*)
2. an acknowledge response (3 bits) transmitted by the target (see *Table 252*)
3. data transfer (33 bits) transmitted by the host (in case of a write) or by the target (in case of a read) (see *Table 253*)

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

Table 251. Packet request

Bit field	Name	Description
0	Start	Must be 1.
1	APnDP	– 0: DP register access - see *Table 250* for a list of DP registers – 1: AP register access - see *Section 36.5: Access port*
2	RnW	– 0: Write request – 1: Read request
4:3	A(3:2)	Address field of the DP or AP registers
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as 1 by the target.

Table 252. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 253. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or read data
32	Parity	Single bit parity of 32 data bits

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial-wire debug protocol, refer to the Arm® Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

36.4 Debug port (DP) registers

Both SW-DP and JTAG-DP access the DP registers listed in *Table 255: DP register map and reset values*.

The debugger accesses the DP registers as follows:

- Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit reset (see *Table 251*). The write data are sent in the data phase.
- To access one of the banked DP registers at address 0x4, the register number must first be written to the DP_SELECTR register at address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the content of the DP_SELECTR register.

Table 254. Debug port registers

Address	A(3:2) value	R/W	Description
0x0	00	R	DP_DIPDR register (see *Section 36.4.1*) Contains the IDCODE for the debug port.
		W	DP_ABORTR register(1) (see *Section 36.4.2*) Aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRLSTATR register.
0x4	01	R/W	If DP_SELECTR.DPBANKSEL = 0, DP_CTRLSTATR register (see *Section 36.4.3*) Controls the DP and provides status information.
			If DP_SELECTR.DPBANKSEL = 1, DP_DLCR register(2) (see *Section 36.4.4*) Controls the operating mode of the SWD data link.
			If DP_SELECTR.DPBANKSEL = 2, DP_TARGETIDR register (see *Section 36.4.5*) Provides target identification information.
			If DP_SELECTR.DPBANKSEL = 3, DP_DLPIDR register(2) (see *Section 36.4.6*) Provides the SWD protocol version.
0x8	10	R	DP_RESENDR register(2) (see *Section 36.4.7*) Returns the value returned by the last AP read or DP_RDBUFFR read. Used in the event of a corrupted read transfer.
		W	DP_SELECTR register (see *Section 36.4.8*) Selects the access port, access port register bank, and DP register at address 0x4.
0xC	11	R	DP_BUFFR register (see *Section 36.4.9*) – Via JTAG-DP - Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation). – Via SW-DP - Contains the result of the preceding AP read access, allowing a new AP access to be avoided.
		W	DP_TARGETSELR register(2) (see *Section 36.4.10*) On a write to DP_TARGETSELR immediately following a line reset sequence, the target is selected if both the following conditions are met: – Bits [31:28] match bits [31:28] in the DP_DLPIDR register. – Bits [27:0] match bits [27:0] in the DP_TARGETIDR register. Writing any other value de-selects the target. Debug tools must write 0xFFFFFFFF to de-select all targets. This is an invalid TARGETID value. All other invalid TARGETID values are reserved.

1. Access to the DP_ABORTR register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.

36.4.1 DP identification register (DP_DPIDR)

Address offset: 0x00

Reset value: 0x5BA0 2477

Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]:** revision code
0x5

Bits 27:20 **PARTNO[7:0]:** part number for the debug port
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN:** minimal debug port (MINDP) implementation
0x0: MINDP not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]:** DP architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]:** JEDEC designer identity code
0x23B: Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

36.4.2 DP abort register (DP_ABORTR)

Address offset: 0x00

Reset value: 0x0000 0000

Write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUN ERR CLR	WD ERR CLR	STK ERR CLR	STK CMP CLR	DAP ABORT
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR:** overrun error clear
0: No effect
1: Clears DP_CTRLSTATR.STICKYORUN bit.

Bit 3 **WDERRCLR:** write data error clear
0: No effect
1: Clears DP_CTRLSTATR.WDATAERR bit.

Bit 2 **STKERRCLR:** sticky error clear
0: No effect
1: Clears DP_CTRLSTATR.STICKYERR bit.

Bit 1 **STKCMPCLR:** sticky compare clear
0: No effect
1: Clears DP_CTRLSTATR.STICKYCMP bit

Bit 0 **DAPABORT:** data AP abort
Aborts current AP transaction if an excessive number of WAIT responses are returned, indicating that the transaction is stalled.
0: No effect
1: Aborts the transaction.

36.4.3 DP control and status register (DP_CTRLSTATR)

Address offset: 0x04

and DP_SELECTR.DPBANKSEL = 0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	TRNCNT[11:4]							
		r	r					r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRNCNT[3:0]				MASKLANE[3:0]				WDATAERR	READOK	STICKYERR	STICKYCMP	TRNMODE[1:0]		STICKYORUN	ORUNDETECT
r	r	r	r	r	r	r	r	r	r	rc_w1	rc_w1	r	r	rc_w1	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGPWRUPACK:** see description in *Section 36.3.6*
0 = DAPCLK gated
1 = DAPCLK enabled

Bit 28 **CDBGPWRUPREQ:** control of DAPCLK enable request signal
0 = Requests DAPCLK gating
1 = Requests DAPCLK enabled

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:12 **TRNCNT[11:0]:** transaction counter
To program a sequence of transactions to incremental addresses via an AP, TRNCNT is loaded with the number of transactions to perform. It is decremented at the successful completion of each transaction.

Bits 11:8 **MASKLANE[3:0]:** masked byte lanes

Indicates the bytes to be masked in pushed-compare and pushed-verify operations (DP_CTRLSTATR.TRNMODE = 1 or 2). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address.

1XXX = includes byte lane 3 in comparisons.

X1XX = includes byte lane 2 in comparisons.

XX1X = includes byte lane 1 in comparisons.

XXX1 = includes byte lane 0 in comparisons.

Bit 7 **WDATAERR:** write data error (read only) in SW-DP

There is a parity or framing error on the data phase of a write, or a write that has been accepted by the DP is then discarded without being submitted to the AP.

This bit is reset by writing 1 to the DP_ABORTR.WDERRCLR bit.

0: No error

1: An error occurred.

Reserved in JTAG-DP.

Bit 6 **READOK:** AP read response (read only) in SW-DP

Indicates the response to the last AP read access.

0: Read not OK

1: Read OK

Reserved in JTAG-DP.

Bit 5 **STICKYERR:** transaction error (read only in SW-DP, R/W in JTAG-DP)

Indicates that an error occurred in an AP transaction.

0: No error

1: An error occurred.

In SW-DP, STICKYERR bit is read only, reset by writing 1 to the DP_ABORTR.STKERRCLR bit.

In JTAG-DP, STICKYERR bit is read, cleared by writing a 1 to it.

Bit 4 **STICKYCMP:** match comparison (read only in SW-DP, R/W in JTAG-DP)

Indicates that a match occurred in a pushed operation.

0: match if TRNMODE = 0x1; no match if TRNMODE = 0x2

1: no match if TRNMODE = 0x1; match if TRNMODE = 0x2

In SW-DP, STICKYCMP bit is read only, reset by writing 1 to the DP_ABORTR.STKCMPCLR bit.

In JTAG-DP, STICKYCMP bit is read, cleared by writing a 1 to it.

Bits 3:2 **TRNMODE[1:0]:** transfer mode for AP write operations

For read operations, this field must be set to 0x0.

0x0: Normal operation

– AP transactions are passed directly to the AP.

0x1: Pushed-verify operation

– The DP stores the write data and performs a read transaction at the target AP address.

– The result of the read is compared with the stored data and if they do not match, the STICKYCMP bit is set.

0x2: Pushed-compare operation

– The DP stores the write data and performs a read transaction at the target AP address.

– The result of the read is compared with the stored data and if they match, the STICKYCMP bit is set.

0x3: reserved

In pushed operation, only the data bytes indicated by the MASKLANE field are included in the compare.

Bit 1 **STICKYORUN:** overrun (read only in SW-DP, R/W in JTAG-DP)

Indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.

0: No overrun

1: An overrun occurred.

In SW-DP, STICKYORUN bit is read only, reset by writing 1 to the DP_ABORTR.ORUNERRCLR bit.

In JTAG-DP, STICKYORUN bit is read, cleared by writing a 1 to it.

Bit 0 **ORUNDETECT:** overrun detection mode enable

0: Overrun detection disabled

1: Overrun detection enabled

In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

36.4.4 DP data link control register (DP_DLCR)

Address offset: 0x04

and DP_SELECTR.DPBANKSEL = 1

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						r	r								

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]:** tristate period for SWDIO
0x0: 1 data bit period
0x1: 2 data bit periods
0x2: 3 data bit periods
0x3: 4 data bit periods

Bits 7:0 Reserved, must be kept at reset value.

36.4.5 DP target identification register (DP_TARGETIDR)

Address offset: 0x04

and DP_SELECTR.DPBANKSEL = 2

Reset value: 0x0497 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]:** target revision
0x0: revision 1

Bits 27:12 **TPARTNO[15:0]:** target part number
0x4970: STM32WLEx

Bits 11:1 **TDESIGNER[10:0]:** target designer JEDEC code.
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

36.4.6 DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x04

and DP_SELECTR.DPBANKSEL = 3

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]:** target instance number
Defines the instance number for this device in a multi-drop system.
0x0: Instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]:** Serial-wire debug protocol version
0x1: Version 2

36.4.7 DP resend register (DP_RESENDR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND[31:0]:** Returns the value that was returned by the last AP read or DP_RDBUFFR read.
Used in the event of a corrupted read transfer.

36.4.8 DP access port select register (DP_SELECTR)

Address offset: 0x08

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:28 **APSEL[3:0]:** access port selection
Selects the access port for the next transaction.
0x0: AP0 - CPU (Cortex-M4) debug access port (AHB-AP)
0x1 to 0xF: reserved

Bits 27:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]:** AP register bank selection
Selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]:** DP register bank selection
Selects the register at address 0x4 of the debug port.
0x0: DP_CTRLSTATR
0x1: DP_DLCR
0x2: DP_TARGETIDR
0x3: DP_DLPIDR
0x4 to 0xF: reserved

36.4.9 DP read buffer register (DP_BUFFR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]:** Contains the value returned by the last AP read access.

The value returned by an AP read access can either be obtained using a second read access to the same address, which initiates a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.

36.4.10 DP target identification register (DP_TARGETSELR)

Address offset: 0x0C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				TPARTNO[15:4]											
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	

Bits 31:28 **TINSTANCE[3:0]:** target instance number

Defines the instance number for the target device in a multi-drop system. These bits must be written with the same value used for DP_DLPIDR.TINSTANCE to select this device.

Bits 27:12 **TPARTNO[15:0]:** target part number

Defines the part number for the target device. These bits must be written with the same value used for DP_TARGETIDR.TPARTNO to select this device.

Bits 11:1 **TDESIGNER[10:0]:** target designer JEDEC code

Defines the JEDEC code for the target device. These bits must be written with the same value used for DP_TARGETIDR.TDESIGNER to select this device.

Bit 0 Reserved, must be kept at reset value.

36.4.11 DP register map and reset values

These registers are not on the CPU memory bus and are only accessed through SW-DP and JTAG-DP debug interface.

The debug port address is 2-bit wide, defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 255. DP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DP_DPIDR	REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION[3:0]				DESIGNER[10:0]											Res.
	Reset value	0	1	0	1	1	0	1	1	1	0	1	0				0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	
0x00	DP_ABORTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	STKCMPCLR	DAPABORT
	Reset value																												0	0	0	0	0
0x04(1)	DP_CTRLSTATR	Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	TRNCNT[11:0]												MASKLANE[3:0]				WDATAERR	READOK	STICKYERR	STICKYCMP	TRNMODE[1:0]		STICKYORUN	ORUNDETECT
	Reset value			0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04(2)	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0								
0x04(3)	DP_TARGETIDR	TREVISION[3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	0	0	0	0	0	1	0	0	1	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
0x04(4)	DP_DLPIDR	TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
	Reset value	0	0	0	0																									0	0	0	1
0x08	DP_RESENDR	RESEND[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 255. DP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	DP_SELECTR	APSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
	Reset value	x	x	x	x																					x	x	x	x	x	x	x	x
0x0C	DP_BUFFR	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DP_TARGETSELR	TINSTANCE[3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

1. DP_SELECTR.DPBANKSEL = 0.
2. DP_SELECTR.DPBANKSEL = 1.
3. DP_SELECTR.DPBANKSEL = 2.
4. DP_SELECTR.DPBANKSEL = 3.

36.5 Access port

As shown in *Figure 372*, there is one access port (AP) attached to the DP:

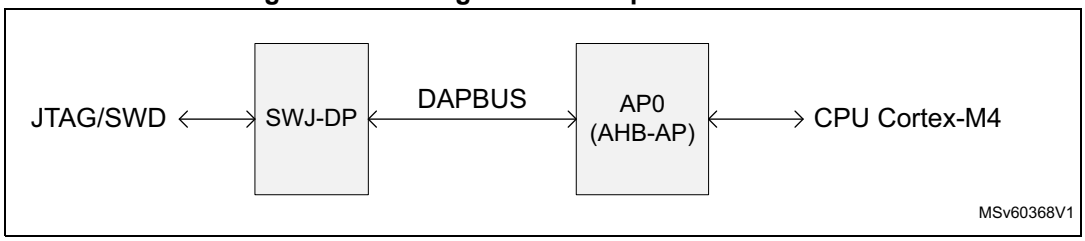
- **AP0**, CPU (Cortex-M4) access port (AHB-AP): enables access to the debug and trace features integrated in the core via its internal AHB bus.

The access port is of MEM-AP type, that is to say the debug and trace component registers are mapped in the address space of the associated debug bus.

AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus.

The AP registers are listed in *Table 257: AP register map and reset values*.

Figure 372. Debug and access port connections



The address of the AP registers is composed as follows:

- Bits [7:4]: content of the APBANKSEL[3:0] field in the DP_SELECTR register (see *Section 36.4.8*)
- Bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 255: DP register map and reset values*) or of the SW-DP packet request (see *Table 251: Packet request*), depending on the debug interface used
- Bits [1:0]: always set to 0

The content of the APSEL[3:0] field of the DP_SELECTR register defines which MEM-AP is being accessed.

Table 256. MEM-AP registers

Address	APBANKSEL	A(3:2)	Name	Description
0x00	0x0	0	AP_CSWR	Control/status word register (see *Section 36.5.1*)
0x04	0x0	1	AP_TAR	Transfer address register (see *Section 36.5.2*) Target address for the bus transaction
0x08	-	-	-	Reserved
0x0C	0x0	3	AP_DRWR	Data read/write register (see *Section 36.5.3*) Access to this register triggers a corresponding transaction on the debug bus to the address in AP_TAR[31:0].
0x10	0x1	0	AP_BD0R	Banked data 0 register (see *Section 36.5.4*) Access to this register triggers a corresponding transaction on the debug bus to the address in Address [31:4] = AP_TAR[31:4], address [3:0] = 0x0.
0x14	0x1	1	AP_BD1R	Banked data 1 register (see *Section 36.5.4*) Access to this register triggers a corresponding transaction on the debug bus to the address in Address [31:4] = AP_TAR[31:4], address [3:0] = 0x4.
0x18	0x1	2	AP_BD2R	Banked data 2 register (see *Section 36.5.4*) Access to this register triggers a corresponding transaction on the debug bus to the address in Address [31:4] = AP_TAR[31:4], address [3:0] = 0x8.
0x1C	0x1	3	AP_BD3R	Banked data 3 register (see *Section 36.5.4*) Access to this register triggers a corresponding transaction on the debug bus to the address in Address [31:4] = AP_TAR[31:4], address [3:0] = 0xC.
0x20	-	-	-	Reserved
0x24 to 0xEC	-	-	-	Reserved
0xF0	-	-	-	Reserved
0xF4	-	-	-	Reserved
0xF8	0xF	2	AP_BASER	Debug base address register (RO) (see *Section 36.5.5*) Base address of the ROM table
0xFC	0xF	3	AP_IDR	Identification register (RO) (see *Section 36.5.6*)

The debugger can access the AP registers as follows:

1. Program in the DP_SELECTR register, the APSEL(3:0] field to choose the AP and the APBANKSEL[3:0] field to select the register bank to be accessed (see *Section 36.4.8*).
2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see *Table 251: Packet request*). The write data is sent in the data phase.

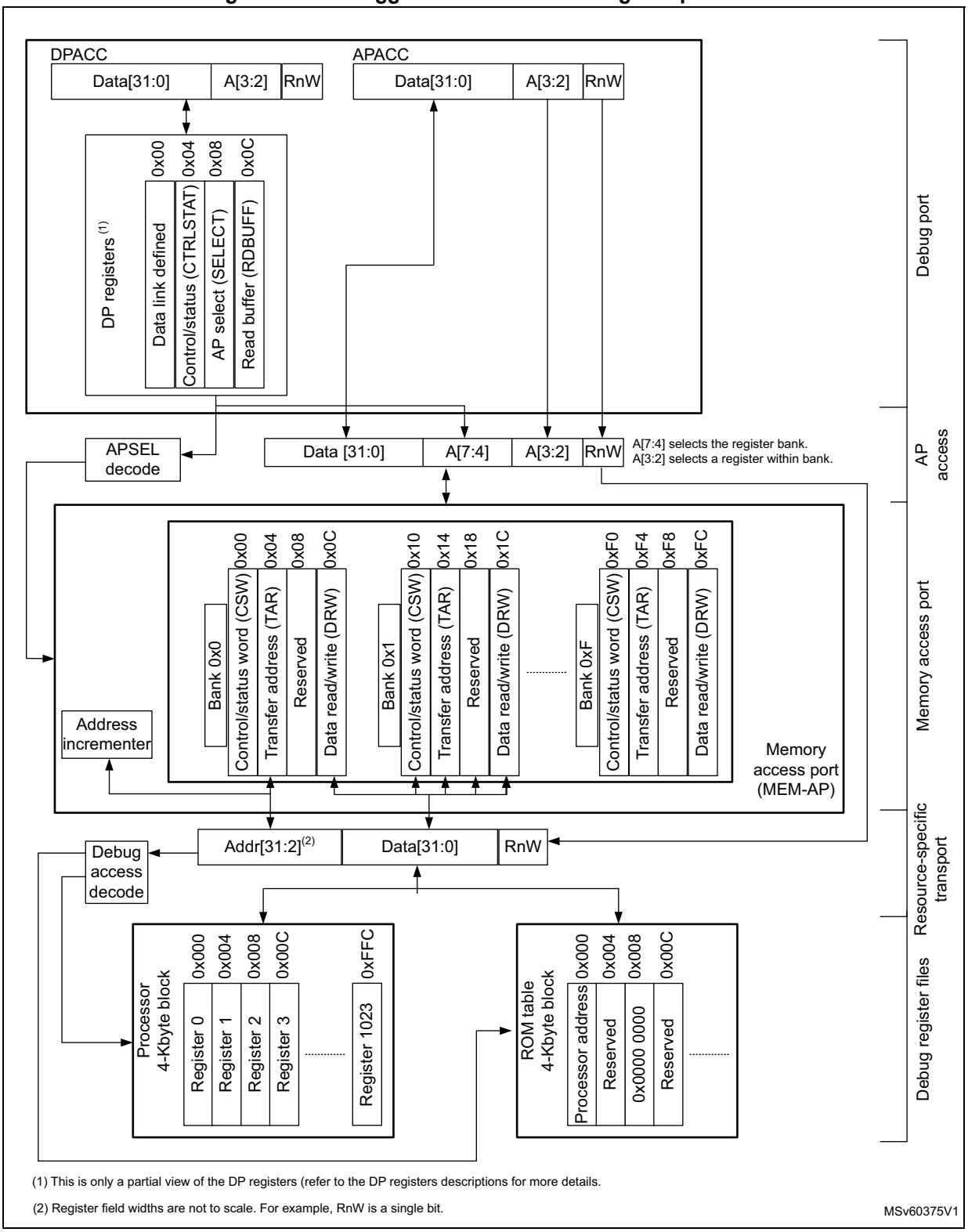
The debugger can access the memory mapped debug component registers through the MEM-AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the AP_TAR register.
2. Program the AP_CSWR register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the AP_DRWR register to initiate a bus transaction at the address held in the AP_TAR register. Alternatively, a read or write to the AP_BDxR register triggers an access to address AP_TAR[31:4] + x (allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register).

Figure 373 shows how the MEM-AP is used to connect the debug port to the debug components (in this example a processor and a ROM table).

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification *[1]*.

Figure 373. Debugger connection to debug components



(1) This is only a partial view of the DP registers (refer to the DP registers descriptions for more details.

(2) Register field widths are not to scale. For example, RnW is a single bit.

MSv60375V1

36.5.1 AP control/status word register (AP_CSWR)

Address offset: 0x00

Reset value: 0x2300 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SPROT	Res.	PROT[4:0]					SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	r		r	r	r	r	r	r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]]		
				r	r	r	r	r	r	r	r		r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT:** secure transfer request
In the AHB-APs, this field sets the protection attribute HPROT[6] of the bus transfer.
0: If SPIDEN is high, secure transfer. If SPIDEN is low, non-secure transfer
1: Non-secure transfer

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **PROT[4:0]:** bus transfer protection
In the AHB-APs, this field sets the protection attributes HPROT[4:0] of the bus transfer.
XXXX0: Instruction fetch
XXXX1: Data access
XXX0X: User mode
XXX1X: Privileged mode
XX0XX: Non-bufferable
XX1XX: Bufferable
X0XXX: Non-cacheable
X1XXX: Cacheable
0XXXX: Non-exclusive
1XXXX: Exclusive

Bit 23 **SPISTATUS:** status of SPIDEN option bit (read only)
This signal determines whether the debugger can access secure memory.
0: Secure AHB transfers blocked
1: Secure AHB transfers allowed

Bits 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]:** barrier support enabled
Defines if memory barrier operation is supported.
0x0: Not supported

Bit 7 **TRINPROG:** transfer in progress (read only)
Indicates if a bus transfer is in progress on the AP.
0x0: No transfer in progress
0x1: Bus transfer in progress

Bit 6 **DEVICEEN:** device enabled (read only)
Defines whether the AP can be accessed.
0x1: AP access enabled

Bits 5:4 **ADDRINC[1:0]:** auto-increment mode

Defines whether AP_TAR address is automatically incremented after a transaction.

0x0: No auto-increment

0x1: Address is incremented by the size in bytes of the transaction (SIZE field).

0x2: Packed transfers enabled

- A 32-bit AP access gives rise to 1 x 32-bit, 2 x 16-bit or 4 x 8-bit bus transactions corresponding to the programmed transaction size.
- The data is packed or unpacked accordingly.

0x3: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]:** size of next memory access transaction

0x0: Byte (8-bit)

0x1: Halfword (16-bit)

0x2: Word (32-bit)

0x3-0x7: reserved

36.5.2 AP transfer address register (AP_TAR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TA[31:0]:** address of current transfer

36.5.3 AP data read/write register (AP_DRWR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TD[31:0]:** data of current transfer

36.5.4 AP banked data registers x (AP_BDxR)

Address offset: 0x10 + 0x04 * x, (x=0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TBD[31:0]:** banked data of current transfer to address AP_TAR.TA
TA + AP_BDnR address [3:2] + 0b00
Auto address incrementing is not performed on AP_BD[3:0]R.
Banked transfers are only supported for word transfers.

36.5.5 AP base address register (AP_BASER)

Address offset: 0xF8

Reset value: 0xE00F F003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[19:4]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[19:0]:** base address (bits 31 to 12) of ROM table for the AP
The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
AP0 CPU (Cortex-M4) AHB-AP: 0xE00FF

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT:** base address register format
1: Arm debug interface v5

Bit 0 **ENTRYPRESENT:** Indicates that debug components are present on the access port bus.
1: Debug components are present

36.5.6 AP identification register (AP_IDR)

Address offset: 0xFC

Reset value: 0x2477 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]:** revision
0x2: Cortex-M4 r0p3

Bits 27:24 **JEDECBANK[3:0]:** JEDEC bank
0x4: Arm

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm

Bit 16 **MEMAP:** memory access port
0x1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]:** AP type
0x11: CPU (Cortex-M4) AHB-AP (AP0)
Others: reserved

36.5.7 AP register map and reset values

These registers are not on the CPU memory bus and are only accessed through SW-DP and JTAG-DP debug interface.

The access port address is 8-bit wide, defined by debug port register DP_SELECTR.APBANKSEL[3:0] field and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 257. AP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP_CSWR	Res.	SPROT	Res.	PROT[4:0]					SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		0		0	0	0	1	1	0												0	0	0	0	0	1	0	0		0	0	0
0x04	AP_TAR	TA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	Reserved	Reserved																															
0x0C	AP_DRWR	TD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 257. AP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	AP_BD0R	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	AP_BD1R	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	AP_BD2R	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	AP_BD3R	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20 to 0xF4	Reserved	Reserved																															
0xF8	AP_BASER	BASEADDR[19:0]																													FORMAT	ENTRYPRESENT	
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1
0xFC	AP_IDR	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	1									0	0	0	1	0	0	0	1

36.6 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as one of the following function:

- watchpoint
- PC sampling trigger
- data address sampling trigger
- data comparator (comparator 1 only)
- clock cycle counter comparator (comparator 0 only)

It also contains counters for:

- clock cycles
- folded instructions
- load store unit (LSU) operations
- sleep cycles
- number of cycles per instruction
- interrupt overhead

A DWT comparator compares the value held in its DWT_COMPxR registers with one of the following:

- a data address
- an instruction address
- a data value
- the cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- one or more DWT data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value
- a watchpoint debug event, on either the PC value or the accessed data address
- a CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual *[5]*.

36.6.1 DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	Res.	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]:** number of comparators implemented (read only)
0x4: Four comparators

Bit 27 **NOTRCPKT:** trace sampling and exception tracing support (read only)
0: Supported

Bit 26 **NOEXTTRIG:** external match signal, CMPMATCH support (read only)
0: Supported

Bit 25 **NOCYCCNT:** cycle counter support (read only)
0: Supported

Bit 24 **NOPRFCNT:** profiling counter support (read only)
0: Supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CYCEVTENA:** enable for POSTCNT underflow event counter packet generation
0: Disabled
1: Enabled

Bit 21 **FOLDEVTENA:** enable for folded instruction counter overflow event generation
0: Disabled
1: Enabled

Bit 20 **LSUEVTENA:** enable for LSU counter overflow event generation
0: Disabled
1: Enabled

Bit 19 **SLEEPEVTENA:** enable for sleep counter overflow event generation
0: Disabled
1: Enabled

Bit 18 **EXCEVTENA:** enable for exception overhead counter overflow event generation
0: Disabled
1: Enabled

Bit 17 **CPIEVTENA:** enable for CPI counter overflow event generation
0: Disabled
1: Enabled

Bit 16 **EXCTRCENA:** enable for exception trace generation
0: Disabled
1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA:** enable for POSTCNT counter used as a timer for periodic PC sample packet generation
0: Disabled
1: Enabled

Bits 11:10 **SYNCTAP[1:0]:** synchronization packet counter tap
Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the synchronization packet rate.
0x0: Disabled. No synchronization packets
0x1: Tap at CYCCNT[24]
0x2: Tap at CYCCNT[26]
0x3: Tap at CYCCNT[28]

Bit 9 **CYCTAP:** Selects the position of the POSTCNT tap on the CYCCNT counter.
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]:** initial value of the POSTCNT counter
Writes to this field are ignored if POSTCNT counter is enabled (CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTPRESET[3:0]:** Reloads value of the POSTCNT counter.

Bit 0 **CYCCNTENA:** enable for CYCCNT counter
0: Disabled
1: Enabled

36.6.2 DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CYCCNT[31:0]:** processor clock cycle counter

36.6.3 DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]:** CPI counter

Counts additional cycles required to execute multi-cycle instructions (except those recorded by DWT_LSUCNTR) and counts any instruction fetch stalls.

36.6.4 DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: exception overhead cycle counter

Counts the number of cycles spent in exception processing.

36.6.5 DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]:** sleep cycle counter
Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

36.6.6 DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]:** load store counter
Counts additional cycles required to execute load and store instructions.

36.6.7 DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]:** folded instruction counter
Increments on each instruction that takes 0 cycles.

36.6.8 DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIASAMPLE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EIASAMPLE[31:0]:** executed Instruction Address sample value
Samples the current value of the program counter.

36.6.9 DWT comparator register x (DWT_COMPxR)

Address offset: 0x020 + 0x010 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COMP[31:0]:** reference value for comparison

36.6.10 DWT mask register x (DWT_MASKxR)

Address offset: 0x024 + 0x010 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **MASK[4:0]:** comparator mask size
Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

36.6.11 DWT function register x (DWT_FUNCTxR)

Address offset: 0x028 + 0x010 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]			
							r					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED:** comparator match (read only)
Indicates if a comparator match has occurred since the register was last read.
0: No match
1: Match occurred

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **DATAVADDR1[3:0]:** When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12 **DATAVADDR0[3:0]:** When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10 **DATAVSIZE[1:0]:** For data value matching, specifies the size of the required data comparison.
0x0: Byte
0x1: Half word
0x2: Word
0x3: reserved

Bit 9 **LNK1ENA:** enable for a second linked comparator
Indicates whether use of a second linked comparator is supported (read only).
0x1: Supported

Bit 8 **DATAVMATCH:** enable for cycle comparison.
0x0: Address comparison
0x1: Data value comparison

Bit 7 **CYCMATCH:** enable for cycle count comparison on comparator 0
This field is reserved for other comparators.
0x0: No cycle count comparison
0x1: Compares DWT_COMP0R with the cycle counter, DWT_CYCCNTR.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **EMITRANGE:** Enables generation of data trace address offset packets (containing data address bits 0 to 15).
0x0: Disabled
0x1: Enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]:** Selection of action to take on comparator match
The meaning of this bit field depends on the setting of the DATAVMATCH and CYCMATCH fields. See *[5]*.

36.6.12 DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

36.6.13 DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** part number bits [7:0]
0x02: DWT part number

36.6.14 DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]:** part number bits [11:8]
0x0: DWT part number

36.6.15 DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** component revision number
0x3: r0p4

Bit 3 **JEDEC:** JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

36.6.16 DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** customer modified
0x0: No customer modifications

36.6.17 DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** component ID bits [7:0]
0x0D: Common ID value

36.6.18 DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]:** component ID bits [11:8]
0x0: Common ID value

36.6.19 DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** component ID bits [23:16]
0x05: Common ID value

36.6.20 DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** component ID bits [31:24]
0xB1: Common ID value

36.6.21 DWT register map

Table 258. DWT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DWT_CTRLR	NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	Res.	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0		0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DWT_CYCCNTR	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DWT_CPICNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x00C	DWT_EXCCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x010	DWT_SLPCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	DWT_LSUCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x018	DWT_FOLDCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DWT_COMP0R	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DWT_MASK0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x028	DWT_FUNCT0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x02C	Reserved	Reserved.																															
0x030	DWT_COMP1R	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DWT_MASK1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x038	DWT_FUNCT1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x03C	Reserved	Reserved.																															

Table 258. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	DWT_COMP2R	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	DWT_MASK2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x048	DWT_FUNCT2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x04C	Reserved	Reserved.																															
0x050	DWT_COMP3R	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	DWT_MASK3R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x058	DWT_FUNCT3R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1 [3:0]				DATAVADDR0 [3:0]				DATAVSIZE [1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION [3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x05C to 0xFCC	Reserved	Reserved.																															
0xFD0	DWT_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved.																															
0xFE0	DWT_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	1	0
0xFE4	DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	0	0	0
0xFE8	DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	1	1	1	0	1	1
0xFEC	DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1

Table 258. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFC	DWT_CIDR3	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 36.7: ROM table* for the register boundary addresses.

36.7 ROM table

The ROM table is a CoreSight component that contains the base addresses of all the CoreSight debug components accessible via the AHB-AP. This table allows a debugger to discover the topology of the CoreSight system automatically.

There is one ROM table in the CPU sub-system. This table is pointed to by the AP_BASER register in the CPU AHB-AP. It contains the base address pointer for the system control space (SCS) registers, which allow the debugger to identify the CPU core, as well as the FPB, DWT, and CTI[(a)].

The ROM table (see *Table 259*) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00FF000 to 0xE00FFFFC.

Table 259. ROM table

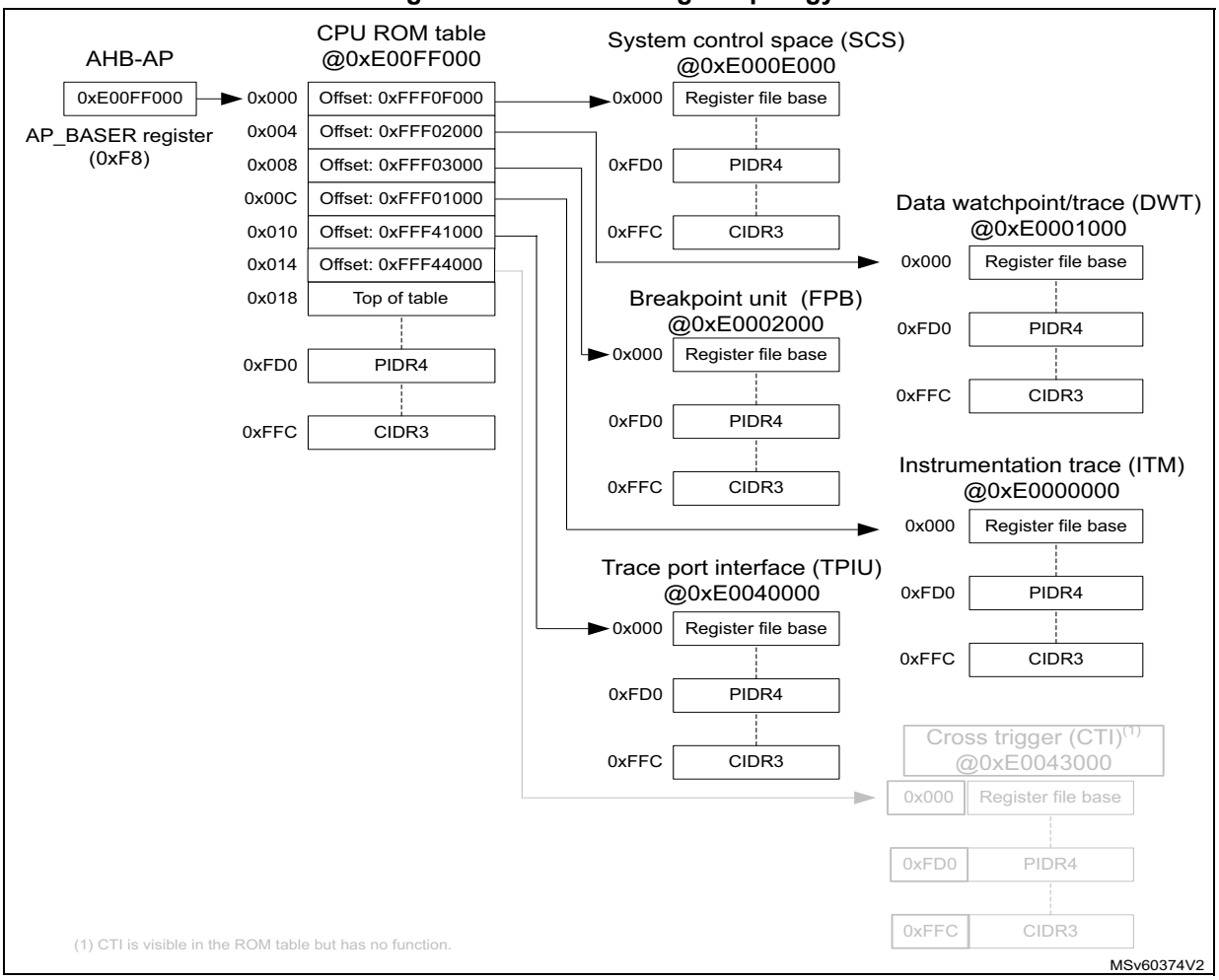
Address in ROM table	Component name	Component base address	Component address offset	Size (bytes)	Entry
0xE00FF000	SCS	0xE000E000	0xFFF0F000	4 K	0xFFF0F003
0xE00FF004	DWT	0xE0001000	0xFFF02000	4 K	0xFFF02003
0xE00FF008	FPB	0xE0002000	0xFFF03000	4 K	0xFFF03003
0xE00FF00C	ITM	0xE0000000	0xFFF01000	4 K	0xFFF01003
0xE00FF010	TPIU	0xE0040000	0xFFF41000	4 K	0xFFF41003
0xE00FF014	CTI[(1)]	0xE0043000	0xFFF44000	4 K	0xFFF44003
0xE00FF018	Top of table	-	-	-	0x00000000
0xE00FF01C to 0xE00FFFC8	Reserved	-	-	-	0x00000000
0xE00FFFCC to 0xE00FFFFC	ROM table registers	-	-	-	See *Table 259*

1. *CTI is visible in the ROM table but has no function.*

The topology for the CoreSight components in the CPU subsystem is shown in *Figure 374*.

a. *CTI is visible in the ROM table but has no function.*

Figure 374. CPU CoreSight topology



36.7.1 ROM memory type register (ROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM:** system memory

1: System memory present on this bus

36.7.2 ROM CoreSight peripheral identity register 4 (ROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

36.7.3 ROM CoreSight peripheral identity register 0 (ROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0097

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** part number bits [7:0]
0x97: STM32WLEx

36.7.4 ROM CoreSight peripheral identity register 1 (ROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: STM32WLEx

36.7.5 ROM CoreSight peripheral identity register 2 (ROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** component revision number
0x0: rev r0p0

Bit 3 **JEDEC:** JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

36.7.6 ROM CoreSight peripheral identity register 3 (ROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** customer modified
0x0: No customer modifications

36.7.7 ROM CoreSight component identity register 0 (ROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** component ID bits [7:0]
0x0D: Common ID value

36.7.8 ROM CoreSight peripheral identity register 1 (ROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]:** component ID bits [11:8]
0x0: Common ID value

36.7.9 ROM CoreSight component identity register 2 (ROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** component ID bits [23:16]
0x05: Common ID value

36.7.10 ROM CoreSight component identity register 3 (ROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** component ID bits [31:24]
0xB1: Common ID value

36.7.11 ROM table register map

Table 260. ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	ROM_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	ROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-0xFDC	Reserved	Reserved.																															
0xFE0	ROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	0	0	1	0	1	1	1
0xFE4	ROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE8	ROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	ROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	ROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1

Table 260. ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFC	ROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 36.7: ROM table* for the register boundary addresses.

36.8 Breakpoint unit (FPB)

The FPB allows the user to set hardware breakpoints. It contains six comparators that monitor the instruction fetch address and two literal address comparators.

If a match occurs, the address is remapped to an address in system memory, defined by the FPB_REMAPR register plus an offset corresponding to the matching comparator.

Alternatively, the instruction comparators can be configured to generate a breakpoint instruction.

36.8.1 FPB control register (FPB_CTRLR)

Address offset: 0x000

Reset value: 0x0000 0260

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE 6	NUM_CODE 5	NUM_CODE 4	NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r	r	r	r	r	r	r	r	r			rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 11:8 **NUM_LIT[3:0]:** number of literal address comparators supported (read only)
0x2: Two literal comparators supported.

Bits 14, 13, 12, 7, 6, 5, 4 **NUM_CODE[6:0]:** number of instruction address comparators supported - least significant bits (read only)
0x6: 6 instruction comparators supported

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY:** write protect key
A write to FPB_CTRLR register is ignored if this bit is not set to 1.

Bit 0 **ENABLE:** FPB enable
0: Disabled
1: Enabled

36.8.2 FPB remap register (FPB_REMAPR)

Address offset: 0x004

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	RMPSPT	REMAP[23:11]												
		r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REMAP[10:0]											Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **RMPSPT:** Flash memory patch remap

Indicates whether Flash memory patch remap is supported (read only).

1: Remapping supported.

Bits 28:5 **REMAP[23:0]:** remap target address

Bits [28:5] of the base address in SRAM to which the FPB remaps the address. The remap base address must be aligned to the number of words required to support the implemented comparators, that is to (NUM_CODE+NUM_LIT) words, with a minimum alignment of 8 words. Because remap is into the SRAM memory region, 0x20000000-0x3FFFFFFF, bits [31:29] of the remap address are 0b001.

Bits 4:0 Reserved, must be kept at reset value.

36.8.3 FPB comparator register x (FPB_COMPxR)

Address offset: 0x008 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REPLACE[1:0]		Res.	COMP[26:14]												
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[13:0]														Res.	ENABLE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:30 **REPLACE[1:0]:** Defines the behavior when a match occurs between the COMP field and the instruction fetch address.

0x0: reserved

0x1: Breakpoint on lower half-word, upper half-word is unaffected.

0x2: Breakpoint on upper half-word, lower half-word is unaffected.

0x3: Breakpoint on both upper and lower half-words

Bit 29 Reserved, must be kept at reset value.

Bits 28:2 **COMP[26:0]:** value to compare with address bits 28:2 of accesses to instruction code memory (0x00000000 to 0x1FFFFFFF)

If a match occurs, the action to be taken is defined by the REPLACE field.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE:** comparator enable

The comparator is only enabled if both this bit and the FPB ENABLE bit in the FPB_CTRLR register are set.

0: Disabled

1: Enabled

36.8.4 FPB CoreSight peripheral identity register 4 (FPB_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** register file size

0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code

0x4: Arm® JEDEC code

36.8.5 FPB CoreSight peripheral identity register 0 (FPB_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** part number bits [7:0]
0x03: FPB part number

36.8.6 FPB CoreSight peripheral identity register 1 (FPB_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]:** part number bits [11:8]
0x0: FPB part number

36.8.7 FPB CoreSight peripheral identity register 2 (FPB_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** component revision number
0x2: r0p3

Bit 3 **JEDEC:** JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

36.8.8 FPB CoreSight peripheral identity register 3 (FPB_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** customer modified
0x0: No customer modifications

36.8.9 FPB CoreSight component identity register 0 (FPB_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** component ID bits [7:0]
0x0D: Common ID value

36.8.10 FPB CoreSight peripheral identity register 1 (FPB_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]:** component ID bits [11:8]
0x0: Common ID value

36.8.11 FPB CoreSight component identity register 2 (FPB_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** component ID bits [23:16]
0x05: Common ID value

36.8.12 FPB CoreSight component identity register 3 (FPB_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** component ID bits [31:24]
0xB1: Common ID value

36.8.13 FPB register map

Table 261. FPB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FPB_CTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	Reset value																		0	0	0	0	0	1	0	0	1	1	0			0	0

Table 261. FPB register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	FPB_REMAPR	Res.	Res.	RMPSP	REMAP[23:0]																								Res.	Res.	Res.	Res.	Res.
	Reset value			1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x008 to 0x024	FPB_COMP0-7R	REPLACE[1:0]		Res.	COMP[26:0]																										Res.	ENABLE	
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x028 to 0xFCC	Reserved	Reserved.																															
0xFD0	FPB_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved.																															
0xFE0	FPB_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	1	1
0xFE4	FPB_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	0	0	0
0xFE8	FPB_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	1	0	1	0	1	1
0xFEC	FPB_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	FPB_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	FPB_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	FPB_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	FPB_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 36.7: ROM table* for the register boundary addresses.

36.9 Instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are three sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The three sources in decreasing order of priority are:

1. Software trace

   Software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

2. Hardware trace

   The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

3. Local timestamping

   The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

36.9.1 ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **STIMULUS[31:0]:** Write data is output on the trace bus as a software event packet.
When reading, bit 0 is a FIFOREADY indicator:
0: Stimulus port buffer is full (or port is disabled).
1: Stimulus port can accept new write data.

36.9.2 ITM trace enable register (ITM_TER)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **STIMENA[31:0]:** enable for stimulus port

Each bit n (31:0) enables the stimulus port associated with the ITM_STIMRn register.

0: Port disabled

1: Port enabled

36.9.3 ITM trace privilege register (ITM_TPR)

Address offset: 0xE00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIVMASK[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIVMASK[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PRIVMASK[31:0]:** Enables unprivileged access to ITM stimulus ports.

Each bit controls eight stimulus ports.

XXX0: Unprivileged access permitted on ports 0 to 7

XXX1: Only privileged access permitted on ports 0 to 7

XX0X: Unprivileged access permitted on ports 8 to 15

XX1X: Only privileged access permitted on ports 8 to 15

X0XX: Unprivileged access permitted on ports 16 to 23

X1XX: Only privileged access permitted on ports 16 to 23

0XXX: Unprivileged access permitted on ports 24 to 31

1XXX: Only privileged access permitted on ports 24 to 31

Note: PRIVMASK is a 32-bit value, the above listed values apply only on the lower 4 bits (PRIVMASK[3:0]), with PRIVMASK[31:4] = 0xXXXXXXX.

36.9.4 ITM trace control register (ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	Res.	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
						rw	rw				r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY:** Indicates whether the ITM is currently processing events (read only).
0: Not busy
1: Busy

Bits 22:16 **TRACEBUSID[6:0]:** identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this field.
Note: Different IDs must be used for each trace source in the system.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TSPRESCALE[1:0]:** local timestamp prescaler
Used with the trace packet reference clock. The possible values are:
0x0: No prescaling
0x1: Divides by 4.
0x2: Divides by 16.
0x3: Divides by 64.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SWOENA:** enable for asynchronous clocking of the timestamp counter (read only)
0: Timestamp counter uses processor clock

Bit 3 **TXENA:** Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: Disabled
1: Enabled

Bit 2 **SYNCENA:** enable for packet transmission synchronization
Note: The debugger setting this bit must also configure the DWT_CTRLR register SYNCTAP field in the DWT for the correct synchronization speed.
0: Disabled
1: Enabled

Bit 1 **TSENA:** enable for local timestamp generation
0: Disabled
1: Enabled

Bit 0 **ITMENA:** ITM enable
0: Disabled
1: Enabled

36.9.5 ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

36.9.6 ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** part number bits [7:0]
0x01: ITM part number

36.9.7 ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]:** part number bits [11:8]
0x0: ITM part number

36.9.8 ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** component revision number
0x3: r0p4

Bit 3 **JEDEC:** JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

36.9.9 ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** customer modified
0x0: No customer modifications

36.9.10 ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** component ID bits [7:0]
0x0D: Common ID value

36.9.11 ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]:** component ID bits [11:8]
0x0: Common ID value

36.9.12 ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** component ID bits [23:16]
0x05: Common ID value

36.9.13 ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** component ID bits [31:24]
0xB1: Common ID value

36.9.14 ITM register map

Table 262. ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 to 0x07C	ITM_STIM0-31R	STIMULUS[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x080	ITM_TER	STIMENA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084 to 0xDCC	Reserved	Reserved.																															
0xE00	ITM_TPR	PRIVMASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE04 to 0xE4C	Reserved	Reserved.																															
0xE80	ITM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	Res.	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0							0	0				0	0	0	0	0
0xE84 to 0xFCC	Reserved	Reserved.																															
0xFD0	ITM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved.																															
0xFE0	ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0xFE4	ITM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	0	0	0

Table 262. ITM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE8	ITM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	1	1	1	0	1	1
0xFEC	ITM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ITM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ITM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	ITM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ITM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

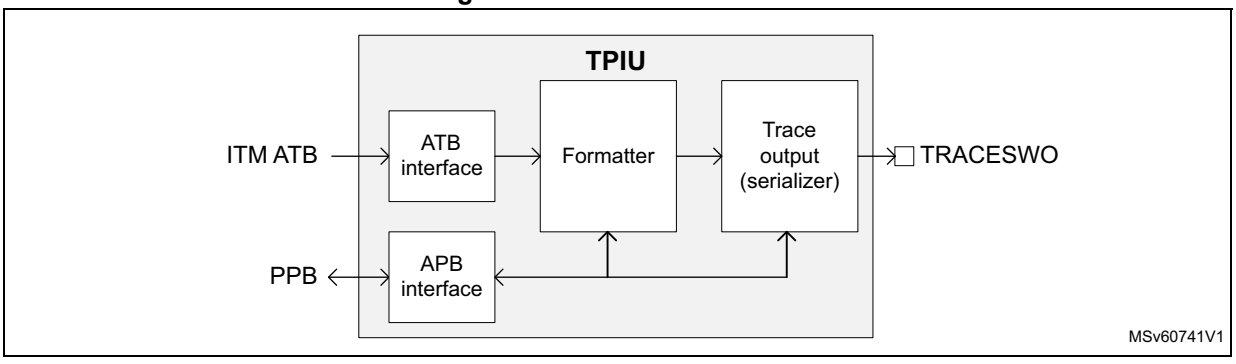
Refer to *Section 36.7: ROM table* for the register boundary addresses.

36.10 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. The TPIU has one ATB slave ports for incoming trace data from the ITM. The trace port is the serial-wire output, TRACESWO.

Figure 375 shows the TPIU architecture.

Figure 375. TPIU architecture



For more information on the TPIU, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual [2.].

36.10.1 TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]:** supported trace port sizes, from 1 to 32 pins
Bit n-1 when set indicates that port size n is supported.
0x0000 000F: Port sizes 1 to 4 supported

36.10.2 TPIU current port size register (TPIU_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PORTSIZE[31:0]:** current trace port size
Bit n-1 when set indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behaviour may result. This register must be modified only when the formatter is stopped.

36.10.3 TPIU asynchronous clock prescaler register (TPIU_ACPR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]:** selects the baud rate for the asynchronous output, TRACESWO
The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

36.10.4 TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TXMODE[1:0]:** selects the protocol used for trace output
0x0: reserved (parallel trace port mode not supported in this device)
0x1: Asynchronous SWO using Manchester encoding
0x2: Asynchronous SWO using NRZ encoding
0x3: reserved

36.10.5 TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNON STOP	TCPRE SENT	FTSTO PPED	FLINPR OG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP:** indicates whether formatter can be stopped or not
1: Formatter cannot be stopped.

Bit 2 **TCPRESENT:** indicates whether the optional TRACECTL output pin is available for use
0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED:** stop request signal received
The formatter received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.
0: Formatter not stopped
1: Formatter stopped

Bit 0 **FLINPROG:** flush in progress
This bit indicates whether a flush on the ATB slave port is in progress and reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.
0: No flush in progress
1: Flush in progress

36.10.6 TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	Res.	Res.	Res.	Res.	Res.	ENFCONT	Res.
							r							rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **TRIGIN:** trigger on trigger in
1: Indicates a trigger in the trace stream when the TRIGIN input is asserted.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** continuous formatting enable
Setting this bit to 0 in SWO mode bypasses the formatter and only ITM/DWT trace is output.
0: Continuous formatting disabled
1: Continuous formatting enabled

Bit 0 Reserved, must be kept at reset value.

36.10.7 TPIU formatter synchronization counter register (TPIU_FSCR)

Address offset: 0x308

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CYCCOUNT[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **CYCCOUNT[12:0]:** enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device

This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

36.10.8 TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]:** sets claim tag bits

Write:
0000: No effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

36.10.9 TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]:** resets claim tag bits
Write:
0000: No effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

36.10.10 TPIU device configuration register (TPIU_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0000 0CA0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWONRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWONRZ:** indicates whether serial-wire output, NRZ, is supported
1: Supported

Bit 10 **SWOMAN:** indicates whether serial-wire output, Manchester encoded format, is supported
1: Supported

Bit 9 **TCLKDATA:** indicates whether trace clock plus data is supported
0: Supported

Bits 8:6 **FIFOSIZE[2:0]:** FIFO size in powers of two
0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT:** indicates relationship between ATB clock and TRACECLKIN
0: Synchronous
1: Asynchronous

Bits 4:0 **MAXNUM[4:0]:** number/type of ATB input port multiplexing
0x0: one input port

36.10.11 TPIU device type identifier register (TPIU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]:** sub-classification
0x1: trace port component

Bits 3:0 **MAJORTYPE[3:0]:** major classification
0x1: trace sink component

36.10.12 TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

36.10.13 TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00A1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** part number bits [7:0]
0x02: TPIU part number

36.10.14 TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]:** part number bits [11:8]
0x0: TPIU part number

36.10.15 TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 004B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** component revision number
0x4: r0p5

Bit 3 **JEDEC:** JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

36.10.16 TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** customer modified
0x0: No customer modifications

36.10.17 TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** component ID bits [7:0]
0x0D: Common ID value

36.10.18 TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** component ID bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]:** component ID bits [11:8]
0x0: Common ID value

36.10.19 TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** component ID bits [23:16]
0x05: Common ID value

36.10.20 TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** component ID bits [31:24]
0xB1: Common ID value

36.10.21 TPIU register map

Table 263. TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TPIU_SSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0x004	TPIU_CSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008 to 0x00C	Reserved	Reserved.																															
0x010	TPIU_ACPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0

Table 263. TPIU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014 to 0x0EC	Reserved	Reserved.																															
0x0F0	TPIU_SPPR	Res.																														TXMODE[1:0]	
	Reset value																															0	1
0x0F4 to 0x2FC	Reserved	Reserved.																															
0x300	TPIU_FFSR	Res.																												FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304	TPIU_FFCR	Res.																							TRIGIN							ENFCONT	Res.
	Reset value																								0							1	
0x308	TPIU_FSCR	Res.																			CYCCOUNT[12:0]												
	Reset value																				0	0	0	0	0	0	1	0	0	0	0	0	0
0x30C to 0xF8C	Reserved	Reserved.																															
0xFA0	TPIU_CLAIMSETR	Res.																												CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.																												CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8 to 0xF	Reserved	Reserved.																															
0xFC8	TPIU_DEVIDR	Res.																				SWONRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT		MAXNUM[4:0]			
	Reset value																					1	1	0	0	1	0	1	0	0	0	0	0
0xFC8	TPIU_DEVTYPER	Res.																								SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFCC	Reserved	Reserved.																															
0xFD0	TPIU_PIDR4	Res.																								F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	1	0	0	0
0xFD4 to 0xFDC	Reserved	Reserved.																															
0xFE0	TPIU_PIDR0	Res																								PARTNUM[7:0]							
	Reset value																									1	0	1	0	0	0	0	1
0xFE4	TPIU_PIDR1	Res																								JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1

Table 263. TPIU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE8	TPIU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	1	0	0	1	0	1	1
0xFEC	TPIU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TPIU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 36.7: ROM table* for the register boundary addresses.

36.11 Microcontroller debug unit (DBGMCU)

DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the debug software) to perform the following tasks:

- Maintain the clock and power to the processor cores when in low-power modes (Sleep, Stop or Standby).
- Maintain the clock and power to the system debug and trace components when in low-power modes.
- Stop the clock to certain peripherals (such as watchdogs, timers, RTC) when the processor core is stopped in debug mode.

DBGMCU registers are not reset by a system reset, only by a power on reset. They are accessible to the debugger via the CPU AHB access port at base address 0xE0042000.

Note: DBGMCU is not a standard CoreSight component, consequently it does not appear in the ROM table.

36.11.1 DBGMCU identity code register (DBGMCU_IDCODER)

Address offset: 0x000

Reset value: 0xXXXX 6497

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]:** revision
For values, refer to the device errata sheet.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]:** device ID
0x497: STM32WLEx

36.11.2 DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_ STANDBY	DBG_ STOP	DBG_ SLEEP
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY:** Allows debug in Standby mode
0: Normal operation. All clocks are disabled and the domain powered down automatically in Standby mode.
1: Automatic clock stop/power down disabled. All active clocks and oscillators continue to run during Standby mode and the domain supply is maintained, allowing full debug capability. On exit from Standby mode, a domain reset is performed.

Bit 1 **DBG_STOP:** Allows debug in Stop mode
0: Normal operation. All clocks are disabled automatically in Stop mode.
1: Automatic clock stop disabled. All active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0 **DBG_SLEEP:** Allows debug in Sleep mode
0: Normal operation. Processor clock is stopped automatically in Sleep mode.
1: Automatic clock stop disabled. Processor clock continue to run, allowing full debug capability.

36.11.3 DBGMCU APB1 peripheral freeze register 1 (DBGMCU_APB1FZR1)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_ LPTIM1 _STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_ I2C3 _STOP	DBG_ I2C2 _STOP	DBG_ I2C1 _STOP	Res.	Res.	Res.	Res.	Res.
rw								rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_ IWDG _STOP	DBG_ WWDG _STOP	DBG_ RTC _STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_ TIM2 _STOP
			rw	rw	rw										rw

Bit 31 **DBG_LPTIM1_STOP:** LPTIM1 stop in CPU debug
0: Normal operation. LPTIM1 continues to operate while CPU is in debug mode.
1: Stop in debug. LPTIM1 is frozen while CPU is in debug mode.

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **DBG_I2C3_STOP:** I2C3 SMBUS timeout stop in CPU debug
0: Normal operation. I2C3 SMBUS timeout continues to operate while CPU is in debug mode.
1: Stop in debug. I2C3 SMBUS timeout is frozen while CPU is in debug mode.

Bit 22 **DBG_I2C2_STOP:** I2C2 SMBUS timeout stop in CPU debug
0: Normal operation. I2C2 SMBUS timeout continues to operate while CPU is in debug mode.
1: Stop in debug. I2C2 SMBUS timeout is frozen while CPU is in debug mode.

Bit 21 **DBG_I2C1_STOP:** I2C1 SMBUS timeout stop in CPU debug
0: Normal operation. I2C1 SMBUS timeout continues to operate while CPU is in debug mode.
1: Stop in debug. I2C1 SMBUS timeout is frozen while CPU is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP:** IWDG stop in CPU debug
0: Normal operation. IWDG continues to operate while CPU is in debug mode.
1: Stop in debug. IWDG is frozen while CPU is in debug mode.

Bit 11 **DBG_WWDG_STOP:** WWDG stop in CPU debug
0: Normal operation. WWDG continues to operate while CPU is in debug mode.
1: Stop in debug. WWDG is frozen while CPU is in debug mode.

Bit 10 **DBG_RTC_STOP:** RTC stop in CPU debug
0: Normal operation. RTC continues to operate while CPU is in debug mode.
1: Stop in debug. RTC is frozen while CPU is in debug mode.

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **DBG_TIM2_STOP:** TIM2 stop in CPU debug
0: Normal operation. TIM2 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM2 is frozen while CPU is in debug mode.

36.11.4 DBGMCU APB1 peripheral freeze register 2 (DBGMCU_APB1FZR2)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_ LPTIM3 _STOP	DBG_ LPTIM2 _STOP	Res.	Res.	Res.	Res.	Res.
									rw	rw					

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DBG_LPTIM3_STOP**: LPTIM3 stop in CPU debug
0: Normal operation. LPTIM3 continues to operate while CPU is in debug mode.
1: Stop in debug. LPTIM3 is frozen while CPU is in debug mode.

Bit 5 **DBG_LPTIM2_STOP**: LPTIM2 stop in CPU debug
0: Normal operation. LPTIM2 continues to operate while CPU is in debug mode.
1: Stop in debug. LPTIM2 is frozen while CPU is in debug mode.

Bits 4:0 Reserved, must be kept at reset value.

36.11.5 DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_ TIM17 _STOP	DBG_ TIM16 _STOP	Res.
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DBG_ TIM1 _STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in CPU debug
0: Normal operation. TIM17 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM17 is frozen while CPU is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in CPU debug
0: Normal operation. TIM16 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM16 is frozen while CPU is in debug mode.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in CPU debug
0: Normal operation. TIM1 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM1 is frozen while CPU is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

36.11.6 DBGMCU register map

Table 264. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBGMCU_ IDCODER	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					0	1	0	0	1	0	0	1	0	1	1	1
0x004	DBGMCU_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value																														0	0	0
0x008-0x038	Reserved	Reserved.																															
0x03C	DBGMCU_ APB1FZR1	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM2_STOP
	Reset value	0								0	0	0									0	0	0										0
0x040	Reserved	Reserved.																															
0x044	DBGMCU_ APB1FZR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM3_STOP	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.
	Reset value																										0	0					
0x048	Reserved	Reserved.																															
0x04C	DBGMCU_ APB2FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0						0											

Refer to *Section 36.11: Microcontroller debug unit (DBGMCU)* for the register boundary addresses.

36.12 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G, 16th March 2015
3. DDI 0461B (ID010111) - Arm® CoreSight™ Trace Memory Controller r0p1 Technical Reference Manual, Issue B, 10 Dec 2010
4. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009
5. DDI 0403D (ID100710) - Arm®v7-M Architecture Reference Manual, Issue E.b, 2 December 2014
6. DDI 0494-2a (ID062813) - Arm® CoreSight™ ETM™-M0+ r0p1 Technical Reference Manual, Issue D, 6 July, 2015
7. DDI 0440C (ID070610) - Arm® CoreSight™ ETM™-M4 r0p1 Technical Reference Manual, Issue C, 29 June 2012

37 Device electronic signature

The device electronic signature is stored in the system memory area of the Flash memory module and can be read using the debug interface or by the CPU.

It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match the characteristics of the microcontroller.

37.1 Device electronic signature registers

37.1.1 Unique device ID register (UID)

The 96-bit unique device identifier is ideally suited:

- for use as serial number (USB string serial number or other end applications)
- for use as part of the security keys to increase the code security in the Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the memory
- during processes such as secure boot

This unique device identifier provides a reference number, unique for a given device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF 7590

Address offset: 0x00

Reset value: 0xXXXX XXXX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]:** X and Y coordinates on the wafer expressed in BCD format

Address offset: 0x04

Reset value: 0xXXXX XXXX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **UID[63:40]:** LOT_NUM[23:0], lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF_NUM[7:0], wafer number (8-bit unsigned number)

Address offset: 0x08

Reset value: 0xXXXX XXXX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** LOT_NUM[55:24], lot number (ASCII encoded)

37.1.2 FLASH size data register (FLASHSIZE)

Base address: 0x1FFF 75E0

Address offset: 0x00

Reset value: 0xXXXX

Note: X is factory-programmed.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **FLASH_SIZE[15:0]:** Flash memory size

These bits indicates the size of the device Flash memory in Kbytes.
As an example, 0x0040 corresponds to 64 Kbytes.

37.1.3 Package data register (PKG)

Base address: 0x1FFF 7500

Address offset: 0x00

Reset value: 0xFFXX

Note: X is either factory- or user-programmed.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[4:0]				
											r	r	r	r	r

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]:** package type
00000: UFBGA73
00010: WLCSP59
01010: UFQFPN48
Others: reserved

37.1.4 IEEE 64-bit unique device ID register (UID64)

A 64-bit unique device identification (UID64) is stored in the Flash memory and can be accessed by the CPUs.

Base address: 0x1FFF 7580

Address offset: 0x00

Reset value: 0xXXXX XXXX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DEVNUM[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEVNUM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DEVNUM[31:0]:** device number
The 32-bit unique device number is a sequential number, different for each individual device.

Address offset: 0x04

Reset value: 0x0080 E115

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STID[7:0]								DEVID[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **STID[23:0]:** company ID
0x0080E1 for STMicroelectronics

Bits 7:0 **DEVID[7:0]:** device ID
0x15

38 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

39 Revision history

Table 265. Document revision history

Date	Revision	Changes
17-Dec-2019	1	Initial release.
9-Jul-2020	2	Updated: – New *Section 4.10.28: Sub-GHz radio SMPS control 2 register (SUBGHZ_SMPSC2R)* – *Figure 21: Clock tree* – *Figure 22: HSE32 clock sources* and *Figure 23: HSE32 TCXO control* – MSICLA[7:0] in *Section 6.4.2: RCC internal clock sources calibration register (RCC_ICSCR)* – Caution in *Section 17.4.6: DMAMUX request line multiplexer* – *Section 16.9: Temperature sensor and internal reference voltage* – TSEL1[3:0] in *Section 17.7.1: DAC control register (DAC_CR)* – TRIM[5:0] in *Section 27.3.2: VREFBUF calibration control register (VREFBUF_CCR)* – New *Table 154: Point on elliptic curve Fp check average computation times* – ADDRERRF and RAMERRF in *Section 23.7.2: PKA status register (PKA_SR)* – CKD[1:0] in *Section 37.4.1: TIM1 control register 1 (TIM1_CR1)* – DTG[7:0] in *Section 37.4.20: TIM1 break and dead-time register (TIM1_BDTR)* – BKCMP2P in *Section 37.4.27: TIM1 alternate function option register 1 (TIM1_AF1)* – *Figure 408: Master/slave connection example with 1 channel only timers* – OC1FE in *Section 38.4.8: TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)* – *Table 307: Output control bit for standard OCx channels* – New *Section 40.3.17: Using timer output as trigger for other timers (TIM16/TIM17)* – IC1M[3:0] in *Section 40.4.7: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)Table 315: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)* – TI1SEL[3:0] in *Section 40.4.19: TIM16 input selection register (TIM16_TISEL)* and *Section 40.4.22: TIM17 input selection register (TIM17_TISEL)* – Note in *Section 26.6: LPTIM interrupts* – PSC[7:01] in *Section 48.7.6: USART guard time and prescaler register (USART_GTPR)* – SBKF in *Section 48.7.9: USART interrupt and status register [alternate] (USART_ISR)*
27-Oct-2020	3	Updated: – *Section 5.2.1: Power-on reset (POR)/power-down reset (PDR) /Brownout reset (BOR)* – Various updates on *Section 7: Hardware semaphore (HSEM)*

Table 265. Document revision history (continued)

Date	Revision	Changes
24-Jun-2021	4	Updated: – Patented technology and errata sheet in the *Introduction* – *Section 3.3.2: Empty check* – OPTVAL in *Section 3.4.2: Option bytes programming* – OPTNV description in *Section 3.10.5: FLASH status register (FLASH_SR)* – *Section 4.1: Sub-GHz radio introduction* – *Section 4.2: Sub-GHz radio main features* – *Section 4.5.5: Generic framing* – *Section 4.1: Sub-GHz radio introduction* – New functionality in *Section 4.6: Sub-GHz radio data buffer* – *Section 4.13.172: Sub-GHz radio receiver gain control register (SUBGHZ_RXGAINCR)* – *Section 4.7.2: Sleep mode* – Note removed in *Set_RfFrequency() command* – *Section 4.10: Sub-GHz radio application configuration* – Danger note removed in *Section 4.13: Sub-GHz radio registers* – *Figure 101: Brownout reset waveform* – PVD naming in *Section 17.6.2: PWR control register 2 (PWR_CR2)* and *Section 17.6.6: Power status register 2 (PWR_SR2)* – First sentence in *Section 40.1.2: System reset* – *External source (HSE32 TXCO)* – First sentence of *Section 30.4: HSEM registers* – Caution in *Section 12.4.4: DMAMUX request line multiplexer* – Note in *Section 12.4.5: DMAMUX request generator* – New note in *Polynomial programmability* – *Figure 30: ADC block diagram* – Formula in *Figure 17.5: RNG processing time* – *Table 119: Interrupt control bits* – *Table 82: CTR mode initialization vector definition* – *Table 85: Initialization of SAES_IVRx registers in CCM mode* – *Section 19.3.4: PKA public key acceleration* – *Table 97: Montgomery multiplication* – *Section 19.5: Example of configurations and processing times* – *Table 122: PKA interrupt requests* – AFIO renamed in *Section 21: General-purpose timer (TIM2)* – New note in *Section 50.4.7: Trigger multiplexer* – Some bit descriptions in *Section 26.7.2: LPTIM interrupt clear register (LPTIM_ICR)* – *Table 430: Error calculation for programmed baud rates at lpuart_ker_ck_pres = 32.768 kHz*

Table 265. Document revision history (continued)

Date	Revision	Changes
19-Apr-2022	5	Updated: – *Section 3.3.1: Flash memory organization* – *Section 4.1: Sub-GHz radio introduction* – *Section 4.2: Sub-GHz radio main features* – *Section 4.5.3: FSK modem* – *Figure 9: Generic packet frames format* – *Section 4.5.7: BPSK framing* – *Table 28: Recommended CAD configuration settings* – *LoRa Set_LoRaSymbTimeout() command* – *Get_RxBufferStatus() command* – New registers in *Section 4.10: Sub-GHz radio registers* – SMPSEN desc in *Section 5.5.8: PWR control register 5 (PWR_CR5)* – *Section 16.3.3: Calibration (ADCAL)* – *Section 16.3.7: Configuring the ADC* – Notes in *Section 16.12.4: ADC configuration register 1 (ADC_CFGR1)* and *Section 16.12.5: ADC configuration register 2 (ADC_CFGR2)* – *Health checks* – *Section 20.3.4: RNG initialization* – *Section 20.5: RNG processing time* – *Section 20.6.2: Validation conditions* – RNDATA desc in *Section 20.7.3: RNG data register (RNG_DR)* – *Section 21.4.16: AES DMA interface* – Note in *Section 23.3.16: Using the break function* – *Section 23.4.4: TIM1 DMA/interrupt enable register (TIM1_DIER)* – Note in *Section 25.3.12: Bidirectional break inputs* – *Section 25.3.13: 6-step PWM generation* – Note on *Section 30.6.1: RTC time register (RTC_TR)* – New *Section 38: Important security notice*

Index

A

C

D

E

F

G

H

I

L

P

R

S

T

U

V

W