RM0440
Reference manual

STM32G4 Series
advanced Arm®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32G4 Series microcontroller memory and peripherals.

The STM32G4 Series is 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 Cortex®-M4 *Technical Reference Manual*.

The STM32G4 Series microcontrollers include ST state-of-the-art patented technology.

Related documents

- Cortex®-M4 Technical Reference Manual, available from: http://infocenter.arm.com
- STM32G4xx datasheets
- STM32F3, STM32F4, STM32G4 and STM32L4 Series Cortex®-M4 programming manual (PM0214)

Contents

List of tables

List of figures

1 Documentation conventions

1.1 General information

The STM32G4 Series devices have an Arm®[a] Cortex®-M4 with FPU core.

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.
- **IAP (in-application programming)**: IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running.
- **ICP (in-circuit programming)**: ICP is the ability to program the Flash memory of a microcontroller using the JTAG protocol, the SWD protocol or the bootloader while the device is mounted on the user application board.
- **Option bytes**: product configuration bits stored in the Flash memory.
- **AHB**: advanced high-performance bus.

1.4 Product category definition

Table 1 gives an overview of memory density versus product line.

The present reference manual describes the superset of features for each product category. Refer to *Table 2* for the list of features per category.

Table 1. STM32G4 Series memory density

Memory density	Category 2	Category 3	Category 4
128 Kbytes	STM32G431 STM32G441 (AES)	STM32G471 STM32G473 STM32G474	-
256 Kbytes	-	STM32G471 STM32G473 STM32G474	-
512 Kbytes	-	STM32G471 STM32G473 STM32G474 STM32G483 (AES) STM32G484 (AES)	STM32G491 STM32G4A1 (AES)

1.5 Availability of peripherals

Table 2: Product specific features summarizes the list of peripherals available in all the STM32G4xx products considering the biggest package in every product.

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

Table 2. Product specific features

Feature	STM32G474/ STM32G484	STM32G473/ STM32G483	STM32G471	STM32G431/ STM32G441	STM32G491/ STM32G4A1
Flash	512/256/128K, Dual bank	512/256/128K, Dual bank	512/256/128K, Dual bank	128/64/32K, single bank	512/256K, single bank
SRAM1	80K, parity check on the first 32K	80K, parity check on the first 32K	80K, parity check on the first 32K	16K, parity check on the whole SRAM1	80K, parity check on the first 32K
SRAM2	16K, no parity check	16K, no parity check	16K, no parity check	6K, no parity check	16K, no parity check
CCM SRAM	32K, parity check on the whole CCM SRAM	32K, parity check on the whole CCM SRAM	16K, parity check on the whole CCM SRAM	10K, parity check on the whole CCM SRAM	16K, parity check on the whole CCM SRAM
CRS	Yes	Yes	Yes	Yes	Yes
DMA	2 DMA controllers: DMA1: 8 channels DMA2: 8 channels	2 DMA controllers: DMA1: 8 channels DMA2: 8 channels	2 DMA controllers: DMA1: 8 channels DMA2: 8 channels	2 DMA controllers: DMA1: 6 channels DMA2: 6 channels	2 DMA controllers: DMA1: 8 channels DMA2: 8 channels
DMAMUX	Yes	Yes	Yes	Yes	Yes
Cordic	Yes	Yes	Yes	Yes	Yes
FMAC	Yes	Yes	Yes	Yes	Yes
RNG	Yes	Yes	Yes	Yes	Yes
AES	Yes (Note)	Yes (Note)	No	Yes (Note)	Yes (Note)
CRC	Yes	Yes	Yes	Yes	Yes
FSMC	Yes	Yes	No	No	No
QUADSPI	Yes	Yes	Yes	No	Yes
ADC	5 x ADC: ADC1/2 can be used in dual mode ADC3/4 can be used in dual mode ADC5 usable only in single mode	5 x ADC: ADC1/2 can be used in dual mode ADC3/4 can be used in dual mode ADC5 usable only in single mode	3 x ADC: ADC1/2 can be used in dual mode ADC3 usable only in single mode	2 x ADC: ADC1/2 can be used in dual mode	3 x ADC: ADC1/2 can be used in dual mode ADC3 usable only in single mode
DAC	7 DAC ch: DAC1_CH1/ DAC1_CH2/ DAC2_CH1: external DAC3_CH1/ DAC3/CH2/ DAC4_CH1/ DAC4_CH2: internal	7 DAC ch: DAC1_CH1/ DAC1_CH2/ DAC2_CH1: external DAC3_CH1/ DAC3/CH2/ DAC4_CH1/ DAC4_CH2: internal	4 DAC ch: DAC1_CH1/ DAC1_CH2: external DAC3_CH1/ DAC3/CH2: internal	4 DAC ch: DAC1_CH1/ DAC1_CH2: external DAC3_CH1/ DAC3/CH2: internal	4 DAC ch: DAC1_CH1/ DAC1_CH2: external DAC3_CH1/ DAC3/CH2: internal
COMP	7 (COMP1..7)	7 (COMP1..7)	4 (COMP1..4)	4 (COMP1..4)	4 (COMP1..4)
OPAMP	6 (OPAMP1..6)	6 (OPAMP1..6)	4 (OPAMP1.2,3,6)	3 (OPAMP1..3)	4 (OPAMP1.2,3,6)
VREFBUF	Yes	Yes	Yes	Yes	Yes

Table 2. Product specific features (continued)

Feature	STM32G474/ STM32G484	STM32G473/ STM32G483	STM32G471	STM32G431/ STM32G441	STM32G491/ STM32G4A1
HRTIM1	Yes	No	No	No	No
Advanced control timers (TIM1/TIM8/ TIM20)	TIM1/8/20	TIM1/8/20	TIM1/8/20	TIM1/8	TIM1/8/20
General purpose timers (TIM2/TIM3/ TIM4/TIM5)	TIM2/3/4/5	TIM2/3/4/5	TIM2/3/4	TIM2/3/4	TIM2/3/4
General purpose timers (TIM15/TIM16/TIM17)	TIM15/16/17	TIM15/16/17	TIM15/16/17	TIM15/16/17	TIM15/16/17
Basic timers (TIM6/TIM7)	TIM6/7	TIM6/7	TIM6/7	TIM6/7	TIM6/7
Low power timer (LPTIM1)	Yes	Yes	Yes	Yes	Yes
Infrared interface (IRTIM)	Yes	Yes	Yes	Yes	Yes
Independent watchdog (IWDG)	Yes	Yes	Yes	Yes	Yes
System window watchdog (WWDG)	Yes	Yes	Yes	Yes	Yes
RTC and TAMP	Yes	Yes	Yes	Yes	Yes
I2C	4 x I2C (I2C1..4)	4 x I2C (I2C1..4)	4 x I2C (I2C1..4)	3 x I2C (I2C1..3)	3 x I2C (I2C1..3)
USART/UART	3 x USART (USART1..3) 2 x UART (UART4,5)	3 x USART (USART1..3) 2 x UART(UART4,5)	3 x USART (USART1..3) 2 x UART(UART4,5)	3 x USART (USART1..3) 1 x UART(UART4)	3 x USART (USART1..3) 2 x UART(UART4,5)
LPUART	1 x LPUART	1 x LPUART	1 x LPUART	1 x LPUART	1 x LPUART
SPI/I2S	4 x SPI/2 x I2S (SPI1..4 - I2S2,3)	4 x SPI/2 x I2S (SPI1..4 - I2S2,3)	4 x SPI/2 x I2S (SPI1..4 -I2S2,3)	3 x SPI/2 x I2S (SPI1..3 - I2S2,3)	3 x SPI/2 x I2S (SPI1..3 - I2S2,3)
SAI	1 x SAI	1 x SAI	1 x SAI	1 x SAI	1 x SAI

Table 2. Product specific features (continued)

Feature	STM32G474/ STM32G484	STM32G473/ STM32G483	STM32G471	STM32G431/ STM32G441	STM32G491/ STM32G4A1
FDCAN	3 x FDCAN (FDCAN1..3)	3 x FDCAN (FDCAN1..3)	2 x FDCAN (FDCAN1,2)	1 x FDCAN (FDCAN1)	2 x FDCAN (FDCAN1,2)
USB device	1 x USB device	1 x USB device	1 x USB device	1 x USB device	1 x USB device
UCPD1	1 x UCPD	1 x UCPD	1 x UCPD	1 x UCPD	1 x UCPD

Note: The AES is available only on STM32G483xx, STM32G484xx, STM32G441x and STM32G4A1 devices.

2 System and memory overview

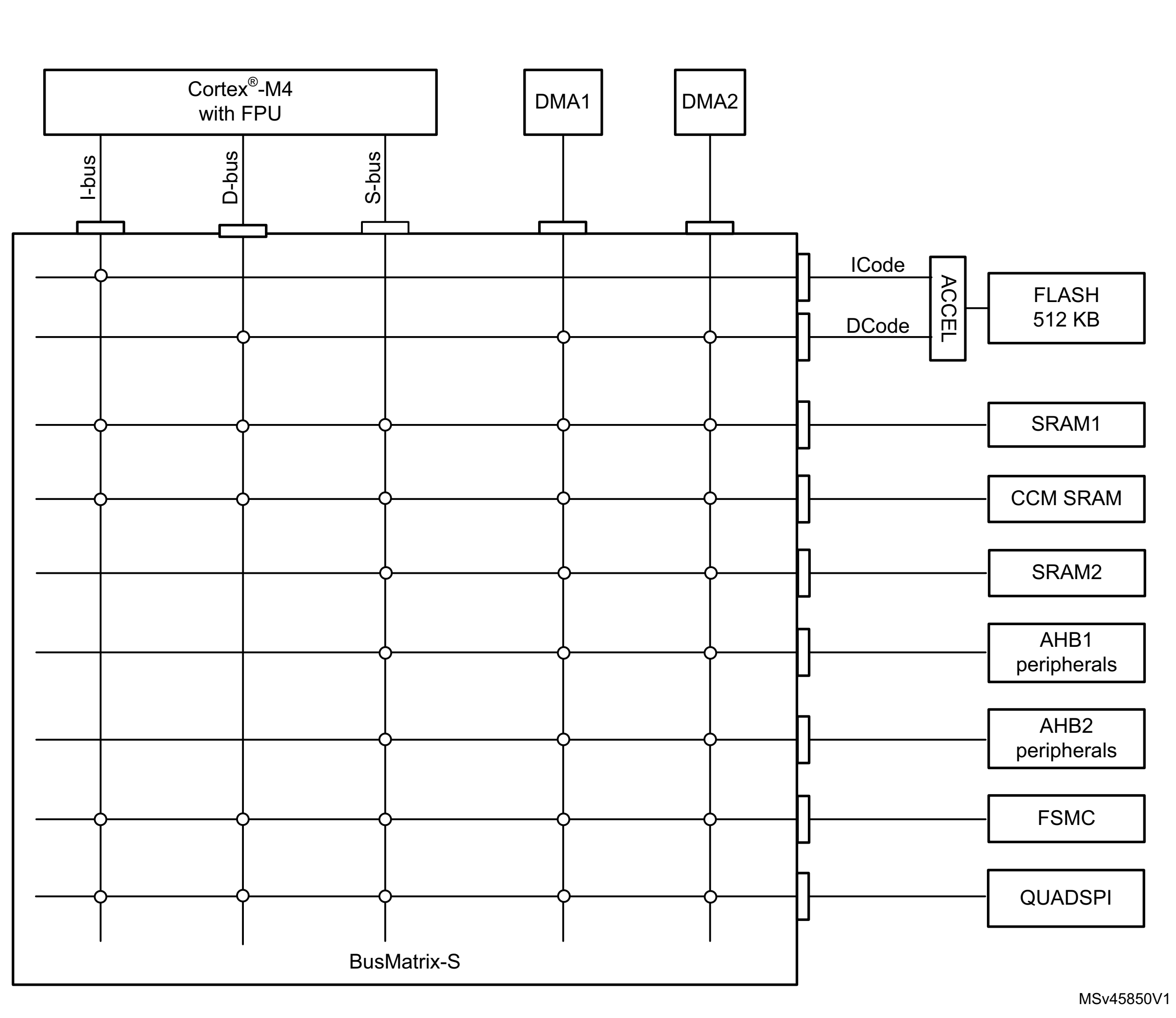
2.1 System architecture

The main system consists of 32-bit multilayer AHB bus matrix that interconnects:

- Up to five masters:
  - Cortex$^{®}$-M4 with FPU core I-bus
  - Cortex$^{®}$-M4 with FPU core D-bus
  - Cortex$^{®}$-M4 with FPU core S-bus
  - DMA1
  - DMA2
- Up to nine slaves:
  - Internal Flash memory on te ICode bus
  - Internal Flash memory on DCode bus
  - Internal SRAM1
  - Internal SRAM2
  - Internal CCM SRAM
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - Flexible static memory controller (FSMC)
  - QUAD SPI memory interface (QUADSPI)

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 *Figure 1*:

Figure 1. System architecture



2.1.1 I-bus

This bus connects the instruction bus of the Cortex®-M4 with FPU core to the BusMatrix. This bus is used by the core to fetch instructions. The target of this bus is a memory containing code (either internal Flash memory, internal SRAMs or external memories through the FSMC or QUADSPI).

2.1.2 D-bus

This bus connects the data bus of the Cortex®-M4 with FPU core to the BusMatrix. This bus is used by the core for literal load and debug access. The target of this bus is a memory containing code (either internal Flash memory, internal SRAMs or external memories through the FSMC or QUADSPI).

2.1.3 S-bus

This bus connects the system bus of the Cortex®-M4 with FPU core to the BusMatrix. This bus is used by the core to access data located in a peripheral or SRAM area. The targets of this bus are the internal SRAM, the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals and the external memories through the QUADSPI or the FSMC.

The CCM SRAM is also accessible on this bus to allow continuous mapping with SRAM1 and SRAM2.

2.1.4 DMA-bus

This bus connects the AHB master interface of the DMA to the BusMatrix.The targets of this bus are the SRAM1, SRAM2 and CCM SRAM, the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals and the external memories through the QUADSPI or the FSMC.

2.1.5 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a Round Robin algorithm. The BusMatrix is composed of up to five masters (CPU AHB, system bus, DCode bus, ICode bus, DMA1, DMA2,) and up to nine slaves (FLASH, SRAM1, SRAM2, CCM SRAM, AHB1 (including APB1 and APB2), AHB2, QUADSPI, and FSMC).

AHB/APB bridges

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

Refer to *Section 2.2.2: Memory map and register boundary addresses on page 82* 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 Flash memory interface). Before using a peripheral you have to enable its clock in the RCC_AHBxENR and the 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 Memory organization

2.2.1 Introduction

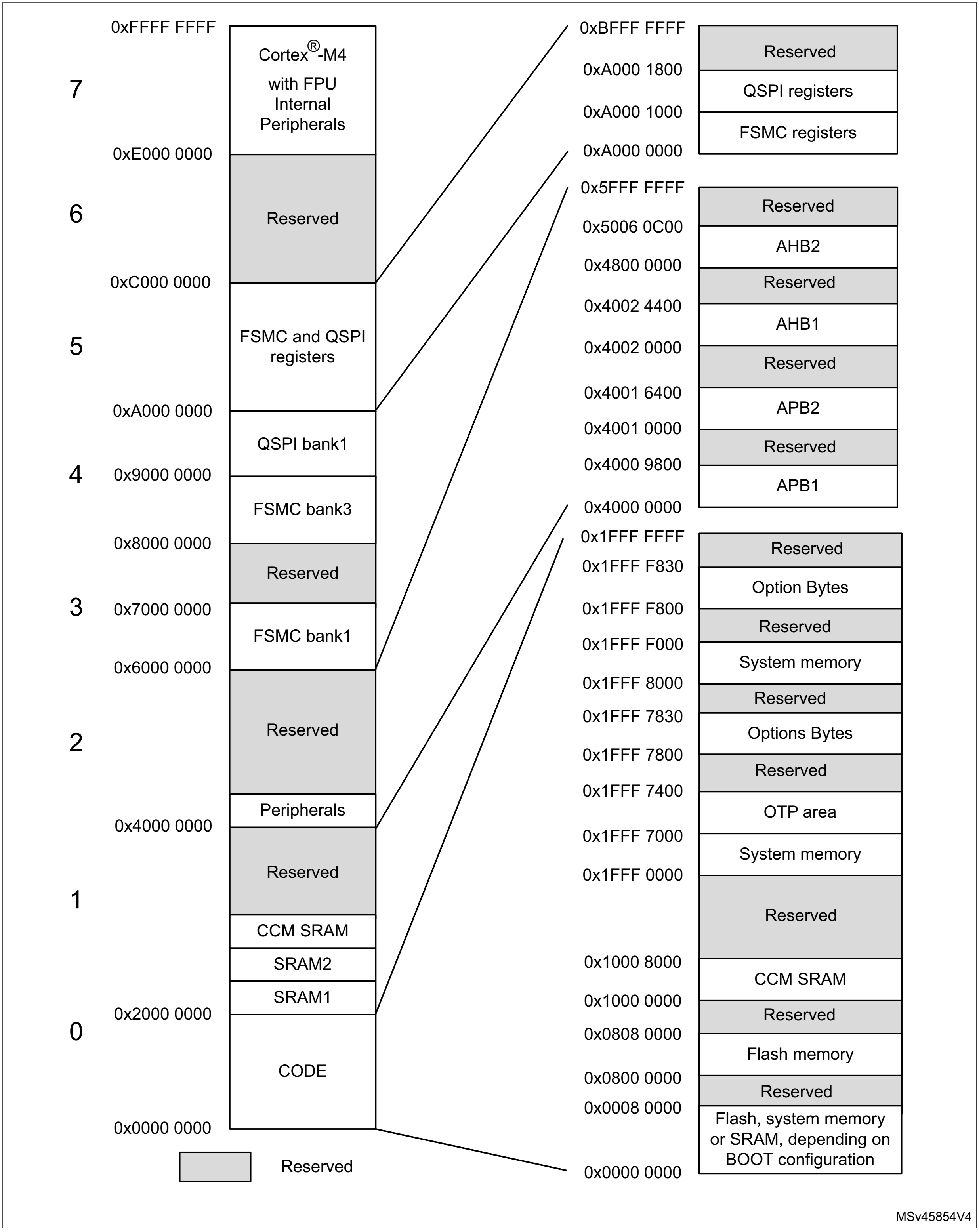
Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

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.

The addressable memory space is divided into eight main blocks, of 512 Mbytes each.

2.2.2 Memory map and register boundary addresses

Figure 2. Memory map



All the memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. For the detailed mapping of available memory and register areas, refer to the following table.

The following table gives the boundary addresses of the peripherals available in the devices.

Table 3. STM32G4 Series memory map and peripheral register boundary addresses[1]

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
-	0xA000 1400 - 0xAFFF FFFF	262 MB	Reserved	-
	0xA000 1000 - 0xA000 13FF	1 KB	QUADSPI control Registers	*Section 20.5.14: QUADSPI register map*
	0xA000 0400 - 0xA000 0FFF	3 KB	Reserved	-
	0xA000 0000 - 0xA000 03FF	1 KB	FSMC	*Section 19.7.8: FMC register map*
AHB2	0x5006 0C00 - 0x5FFF FFFF	256MB	Reserved	-
	0x5006 0800 - 0x5006 0BFF	1 KB	RNG	*Section 26.7.4: RNG register map*
	0x5006 0400 - 0x5006 07FF	1 KB	Reserved	-
	0x5006 0000 - 0x5006 03FF	1 KB	AES	*Section 34.7.18: AES register map*
	0x5000 1800 - 0x5005 FFFF	377 KB	Reserved	-
	0x5000 1400 - 0x5000 17FF	1 KB	DAC4	*Section 22.7.24: DAC register map*
	0x5000 1000 - 0x5000 13FF	1 KB	DAC3	*Section 22.7.24: DAC register map*
	0x5000 0C00 - 0x5000 0FFF	1 KB	DAC2	*Section 22.7.24: DAC register map*
	0x5000 0800 - 0x5000 0BFF	1 KB	DAC1	*Section 22.7.24: DAC register map*
	0x5000 0400 - 0x5000 07FF	1 KB	ADC3 - ADC4 - ADC5	*Section 21.8: ADC register map*
	0x5000 0000 - 0x5000 03FF	1 KB	ADC1 - ADC2	*Section 21.8: ADC register map*
	0x4800 1C00 - 0x4FFF FFFF	127 MB	Reserved	-
	0x4800 1800 - 0x4800 1BFF	1 KB	GPIOG	*Section 9.4.12: GPIO register map*
	0x4800 1400 - 0x4800 17FF	1 KB	GPIOF	*Section 9.4.12: GPIO register map*
	0x4800 1000 - 0x4800 13FF	1 KB	GPIOE	*Section 9.4.12: GPIO register map*
	0x4800 0C00 - 0x4800 0FFF	1 KB	GPIOD	*Section 9.4.12: GPIO register map*
	0x4800 0800 - 0x4800 0BFF	1 KB	GPIOC	*Section 9.4.12: GPIO register map*
	0x4800 0400 - 0x4800 07FF	1 KB	GPIOB	*Section 9.4.12: GPIO register map*
	0x4800 0000 - 0x4800 03FF	1 KB	GPIOA	*Section 9.4.12: GPIO register map*

Table 3. STM32G4 Series memory map and peripheral register boundary addresses[1] (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB1	0x4002 3400 - 0x47FF FFFF	127 MB	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 KB	CRC	*Section 16.4.6: CRC register map*
	0x4002 2400 - 0x4002 2FFF	3 KB	Reserved	-
	0x4002 2000 - 0x4002 23FF	1 KB	Flash interface	*Section 5.7.14: FLASH register map*
	0x4002 1400 - 0x4002 1FFF	3 KB	FMAC	*Section 18.4.9: FMAC register map*
	0X4002 1000 - 0x4002 13FF	1 KB	RCC	*Section 7.4.31: RCC register map*
	0x4002 0C00 - 0x4002 0FFF	1 KB	CORDIC	*Section 17.4.4: CORDIC register map*
	0x4002 0800 - 0x4002 0BFF	1 KB	DMAMUX	*Section 13.6.7: DMAMUX register map*
	0x4002 0400 - 0x4002 07FF	1 KB	DMA 2	*Section 12.6.7: DMA register map*
	0x4002 0000 - 0x4002 03FF	1 KB	DMA 1	*Section 12.6.7: DMA register map*
APB2	0x4001 7800 - 0x4001 FFFF	2 KB	Reserved	-
	0x4001 6800 - 0x4001 77FF	3 KB	HRTIM	*Section 27.5.82: HRTIM register map*
	0x4001 5800 - 0x4001 67FF	4 KB	Reserved	-
	0x4001 5400 - 0x4001 57FF	1 KB	SAI1	*Section 40.6.19: SAI register map*
	0x4001 5000 - 0x4001 53FF	1 KB	TIM20	*Section 28.6.31: TIMx register map*
	0x4001 4C00 - 0x4001 4FFF	1 KB	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17	*Section 30.8.22: TIM16/TIM17 register map*
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16	*Section 30.8.22: TIM16/TIM17 register map*
	0x4001 4000 - 0x4001 43FF	1 KB	TIM15	*Section 30.7.23: TIM15 register map*
	0x4001 3C00 - 0x4001 3FFF	1 KB	SPI4	*Section 39.9.10: SPI/I2S register map*
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1	*Section 37.8.15: USART register map*
	0x4001 3400 - 0x4001 37FF	1 KB	TIM8	*Section 28.6.31: TIMx register map*
	0x4001 3000 - 0x4001 33FF	1 KB	SPI1	*Section 39.9.10: SPI/I2S register map*
	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1	*Section 28.6.31: TIMx register map*
	0x4001 0800 - 0x4001 2BFF	9 KB	Reserved	-
	0x4001 0400 - 0x4001 07FF	1 KB	EXTI	*Section 15.5.13: EXTI register map*
	0x4001 0300 - 0x4001 03FF	1 KB	OPAMP	*Section 25.5.13: OPAMP register map*
	0x4001 0200 - 0x4001 02FF		COMP	*Section 24.6.2: COMP register map*
	0x4001 0030 - 0x4001 01FF		VREFBUF	*Section 23.4.3: VREFBUF register map*
	0x4001 0000 - 0x4001 0029		SYSCFG	*Section 10.2.11: SYSCFG register map*

Table 3. STM32G4 Series memory map and peripheral register boundary addresses[1] (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 AFFE - 0x4000 FFFF	23 KB	Reserved	-
	0x4000 AC00 - 0x4000 AFFF	1 KB	FDCANs Message RAM	*Section 44.4.38: FDCAN register map*
	0x4000 A800 - 0x4000 ABFF	1 KB		
	0x4000 A400 - 0x4000 A7FF	1 KB		
	0x4000 A000 - 0x4000 A3FF	1 KB	UCPD1	*Section 46.7.15: UCPD register map*
	0x4000 8800 - 0x4000 9FFF	6 KB	Reserved	-
	0x4000 8400 - 0x4000 87FF	1 KB	I2C4	*Section 41.7.12: I2C register map*
	0x4000 8000 - 0x4000 83FF	1 KB	LPUART1	*Section 38.7.13: LPUART register map*
	0x4000 7C00 - 0x4000 7FFF	1 KB	LPTIM1	*Section 32.7.10: LPTIM register map*
	0x4000 7800 - 0x4000 7BFF	1 KB	I2C3	*Section 41.7.12: I2C register map*
	0x4000 7400 - 0x4000 77FF	1 KB	Reserved	-
	0x4000 7000 - 0x4000 73FF	1 KB	PWR	*Section 6.4.23: PWR register map and reset value table*
	0x4000 6C00 - 0x4000 6FFF	1 KB	FDCAN3	*Section 44.4.38: FDCAN register map*
	0x4000 6800 - 0x4000 6BFF	1 KB	FDCAN2	*Section 44.4.38: FDCAN register map*
	0x4000 6400 - 0x4000 67FF	1 KB	FDCAN1	*Section 44.4.38: FDCAN register map*
	0x4000 6000 - 0x4000 63FF	1 KB	USB SRAM 1 Kbyte	-
	0x4000 5C00 - 0x4000 5FFF	1 KB	USB device FS	*Section 45.6.3: USB register map*
	0x4000 5800 - 0x4000 5BFF	1 KB	I2C2	*Section 41.7.12: I2C register map*
	0x4000 5400 - 0x4000 57FF	1 KB	I2C1	*Section 41.7.12: I2C register map*
	0x4000 5000 - 0x4000 53FF	1 KB	UART5	*Section 37.8.15: USART register map*
	0x4000 4C00 - 0x4000 4FFF	1 KB	UART4	*Section 37.8.15: USART register map*
	0x4000 4800 - 0x4000 4BFF	1 KB	USART3	*Section 37.8.15: USART register map*
	0x4000 4400 - 0x4000 47FF	1 KB	USART2	*Section 37.8.15: USART register map*

Table 3. STM32G4 Series memory map and peripheral register boundary addresses[1] (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1 Cont.	0x4000 4000 - 0x4000 43FF	1 KB	Reserved	-
	0x4000 3C00 - 0x4000 3FFF	1 KB	SPI3/I2S3	*Section 39.9.10: SPI/I2S register map*
	0x4000 3800 - 0x4000 3BFF	1 KB	SPI2/I2S2	*Section 39.9.10: SPI/I2S register map*
	0x4000 3400 - 0x4000 37FF	1 KB	Reserved	-
	0x4000 3000 - 0x4000 33FF	1 KB	IWDG	*Section 42.4.6: IWDG register map*
	0x4000 2C00 - 0x4000 2FFF	1 KB	WWDG	*Section 43.5.4: WWDG register map*
	0x4000 2800 - 0x4000 2BFF	1 KB	RTC & BKP Registers	*Section 35.6.21: RTC register map*
	0x4000 2400 - 0x4000 27FF	1 KB	TAMP	*Section 36.6.9: TAMP register map*
	0x4000 2000 - 0x4000 23FF	1 KB	CRS	*Section 8.7.5: CRS register map*
	0x4000 1C00 - 0x4000 1FFF	1 KB	Reserved	-
	0x4000 1800 - 0x4000 1BFF	1 KB	Reserved	-
	0x4000 1400 - 0x4000 17FF	1 KB	TIM7	*Section 29.5.31: TIMx register mapSection 31.4.9: TIMx register map*
	0x4000 1000 - 0x4000 13FF	1 KB	TIM6	*Section 31.4.9: TIMx register map*
	0x4000 0C00 - 0x4000 0FFF	1 KB	TIM5	*Section 29.5.31: TIMx register map*
	0x4000 0800 - 0x4000 0BFF	1 KB	TIM4	*Section 29.5.31: TIMx register map*
	0x4000 0400 - 0x4000 07FF	1 KB	TIM3	*Section 29.5.31: TIMx register map*
	0x4000 0000 - 0x4000 03FF	1 KB	TIM2	*Section 29.5.31: TIMx register map*

1. Refer to *Table 1: STM32G4 Series memory density*, *Table 2: Product specific features* and to the device datasheets for the GPIO ports and peripherals available on your device. the memory area corresponding to unavailable GPIO ports or peripherals are reserved (highlighted in gray).

2.3 Bit banding

The Cortex®-M4 with FPU memory 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.

In the STM32G4 Series devices both the peripheral registers and the SRAM are mapped to a bit-band region, so that single bit-band write and read operations are allowed. The operations are only available for Cortex®-M4 with FPU accesses, and not from other bus masters (e.g. DMA).

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:

bit_word_addr = *bit_band_base* + (*byte_offset* x 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 0x20000300 to the alias region:

0x22006008 = 0x22000000 + (0x300*32) + (2*4)

Writing to address 0x22006008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM1 address 0x20000300.

Reading address 0x22006008 returns the value (0x01 or 0x00) of bit 2 of the byte at SRAM1 address 0x20000300 (0x01: bit set; 0x00: bit reset).

2.4 Embedded SRAM

The STM32G4 Series category 3 devices feature up to 128 Kbytes SRAM:

- 80 Kbytes SRAM1 (mapped at address 0x2000 0000)
- 16 Kbytes SRAM2 (mapped at address 0x2001 4000)
- 32 Kbytes CCM SRAM (mapped at address 0x1000 0000 and end of SRAM2)

The STM32G4 Series category 4 devices feature up to 112 Kbytes SRAM:

- 80 Kbytes SRAM1 (mapped at address 0x2000 0000)
- 16 Kbytes SRAM2 (mapped at address 0x2001 4000)
- 16 Kbytes CCM SRAM (mapped at address 0x1000 0000 and end of SRAM2)

The STM32G4 Series category 2 devices feature up to 32 Kbytes SRAM:

- 16 Kbytes SRAM1 (mapped at address 0x2000 0000)
- 6 Kbytes SRAM2 (mapped at address 0x2000 4000)
- 10 Kbytes CCM SRAM (mapped at address 0x1000 0000 and end of SRAM2)

These SRAM can be accessed as bytes, half-words (16 bits) or full words (32 bits). These memories can be addressed at maximum system clock frequency without wait state and thus by both CPU and DMA.

The CPU can access the SRAM1 through the system bus or through the ICode/DCode buses when boot from SRAM1 is selected or when physical remap is selected (*Section 10.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller). To get the maximum performance on SRAM1 execution, physical remap should be selected (boot or software selection).

CCM SRAM is mapped at address 0x1000 0000.

Execution can be performed from CCM SRAM with maximum performance without any remap thanks to access through ICode bus.
The CCM SRAM is aliased at address following the end of SRAM2 (0x2000 5800 for category 2 devices, 0x2001 8000 for category 3 devices, 0x2001 8000 for category 4 devices), offering a continuous address space with the SRAM1 and SRAM2.

2.4.1 Parity check

On the Category 3 and Category 4 devices, a parity check is implemented on the first 32 Kbytes of SRAM1 and on the whole CCM SRAM.

On the Category 2 devices, a parity check is implemented on the whole SRAM1 and CCM SRAM.

The user can enable the parity check using the option bit SRAM_PE in the user option byte (refer to *Section 3.4.1: Option bytes description*).

The data bus width is 36 bits because 4 bits are available for parity check (1 bit per byte) in order to increase memory robustness, as required for instance by Class B or SIL norms.

The parity bits are computed and stored when writing into the SRAM. Then, they are automatically checked when reading. If one bit fails, an NMI is generated. The same error can also be linked to the BRK_IN Break input of TIM1/TIM8/TIM15/TIM16/TIM17/TIM20, and to hrtim_sys_flt with the SPL control bit in the *Section 10.2.8: SYSCFG configuration register 2 (SYSCFG_CFGR2)*. The SRAM Parity Error flag (SPF) is available in the *Section 10.2.8: SYSCFG configuration register 2 (SYSCFG_CFGR2)*.

Note: When enabling the SRAM parity check, it is advised to initialize by software the whole SRAM memory at the beginning of the code, to avoid getting parity errors when reading non-initialized locations.

2.4.2 CCM SRAM Write protection

The CCM SRAM can be write protected with a page granularity of 1 Kbyte.

Table 4. CCM SRAM organization

Page number	Start address	End address
Page 0	0x1000 0000	0x1000 03FF
Page 1	0x1000 0400	0x1000 07FF
Page 2	0x1000 0800	0x1000 0BFF
Page 3	0x1000 0C00	0x1000 0FFF
Page 4	0x1000 1000	0x1000 13FF
Page 5	0x1000 1400	0x1000 17FF
Page 6	0x1000 1800	0x1000 1BFF
Page 7	0x1000 1C00	0x1000 1FFF
Page 8	0x1000 2000	0x1000 23FF
Page 9	0x1000 2400	0x1000 27FF
Page 10[1]	0x1000 2800	0x1000 2BFF

Table 4. CCM SRAM organization (continued)

Page number	Start address	End address
Page 11[1]	0x1000 2C00	0x1000 2FFF
Page 12[1]	0x1000 3000	0x1000 33FF
Page 13[1]	0x1000 3400	0x1000 37FF
Page 14[1]	0x1000 3800	0x1000 3BFF
Page 15[1]	0x1000 3C00	0x1000 3FFF
Page 16[2]	0x1000 4000	0x1000 43FF
Page 17[2]	0x1000 4400	0x1000 47FF
Page 18[2]	0x1000 4800	0x1000 4BFF
Page 19[2]	0x1000 4C00	0x1000 4FFF
Page 20[2]	0x1000 5000	0x1000 53FF
Page 21[2]	0x1000 5400	0x1000 57FF
Page 22[2]	0x1000 5800	0x1000 5BFF
Page 23[2]	0x1000 5C00	0x1000 5FFF
Page 24[2]	0x1000 6000	0x1000 63FF
Page 25[2]	0x1000 6400	0x1000 67FF
Page 26[2]	0x1000 6800	0x1000 6BFF
Page 27[2]	0x1000 6C00	0x1000 6FFF
Page 28[2]	0x1000 7000	0x1000 73FF
Page 29[2]	0x1000 7400	0x1000 77FF
Page 30[2]	0x1000 7800	0x1000 7BFF
Page 31[2]	0x1000 7C00	0x1000 7FFF

1. Available on Category 3 and Category 4 devices only.
2. Available on Category 3 devices only.

The write protection can be enabled in *Section 10.2.9: SYSCFG CCM SRAM write protection register (SYSCFG_SWPR)* in the SYSCFG block. This is a register with write ‘1’ once mechanism, which means by writing ‘1’ on a bit it sets up the write protection for that page of SRAM and it can be removed/cleared by a system reset only.

2.4.3 CCM SRAM read protection

The CCMSRAM is protected with the Read protection (RDP). Refer to *Section 3.5.1: Read protection (RDP)* for more details.

2.4.4 CCM SRAM erase

The CCMSRAM can be erased with a system reset using the option bit CCMSRAM_RST in the user option byte (refer to *Section 3.4.1: Option bytes description*).

The CCM SRAM erase can also be requested by software by setting the bit CCMSR in the *Section 10.2.7: SYSCFG CCM SRAM control and status register (SYSCFG_SCSR)*.

2.5 Flash memory overview

The Flash memory is composed of two distinct physical areas:

- The main Flash memory block. It contains the application program and user data if necessary.
- The information block. It is composed of three parts:
  - Option bytes for hardware and memory protection user configuration.
  - System memory that contains the ST proprietary code.
  - OTP (one-time programmable) area

The Flash interface implements instruction access and data access based on the AHB protocol. It also implements the logic necessary to carry out the Flash memory operations (program/erase) controlled through the Flash registers. Refer to *Section 3: Embedded Flash memory (FLASH) for category 3 devices*, *Section 5: Embedded Flash memory (FLASH) for category 2 devices* and *Section 5: Embedded Flash memory (FLASH) for category 2 devices* for more details.

2.6 Boot configuration

2.6.1 Boot configuration

Three different boot modes can be selected through the BOOT0 pin or the nBOOT0 bit into the FLASH_OPTR register (if the nSWBOOT0 bit is cleared into the FLASH_OPTR register), and nBOOT1 bit in FLASH_OPTR register, as shown in the following table.

Table 5. Boot modes

BOOT_ LOCK	nBOOT1 FLASH_OPTR[23]	nBOOT0 FLASH_OPTR[27]	BOOT0 pin PB8	nSWBOOT0 FLASH_OPTR[26]	Boot Memory Space Alias
1	X	X	X	X	Main Flash memory
0	X	X	0	1	Main Flash memory is selected as boot area
0	X	1	X	0	Main Flash memory is selected as boot area
0	0	X	1	1	Embedded SRAM1 is selected as boot area
0	0	0	X	0	Embedded SRAM1 is selected as boot area
0	1	X	1	1	System memory is selected as boot area
0	1	0	X	0	System memory is selected as boot area

The values on both BOOT0 pin (coming from the pin or the option bit) and nBOOT1 bit are latched on the 4th edge of the internal startup clock source after reset release. It is up to the user to set nBOOT1 and BOOT0 to select the required boot mode.

The BOOT0 pin or user option bit (depending on the nSWBOOT0 bit value in the FLASH_OPTR register), and nBOOT1 bit are also re-sampled when exiting from Standby mode. Consequently, they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value 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 memory or SRAM1 is accessible as follows:

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x0800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF 0000).
- Boot from the embedded SRAM1: the SRAM1 is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x2000 0000).

PB8/BOOT0 GPIO is configured in:

- Input mode during the complete reset phase if the option bit nSWBOOT0 is set into the FLASH_OPTR register and then switches automatically in analog mode after reset is released (BOOT0 pin).
- Input mode from the reset phase to the completion of the option byte loading if the bit nSWBOOT0 is cleared into the FLASH_OPTR register (BOOT0 value coming from the option bit). It switches then automatically to the analog mode even if the reset phase is not complete.

Note: When the device boots from SRAM, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and the offset register.

When booting from the main Flash memory, the application software can either boot from bank 1 or from bank 2 (only for category 3 devices). By default, boot from bank 1 is selected. To select boot from Flash memory bank 2, set the BFB2 bit in the user option bytes. When this bit is set and the boot pins are in the boot from main Flash memory configuration, the device boots from system memory, and the boot loader jumps to execute the user application programmed in Flash memory bank 2. For further details, please refer to AN2606.
See *Table 13: Access status versus protection level and execution modes* for bootloader function for different RDP levels

Forcing boot from user Flash memory

Regardless the boot configuration it is possible to force booting from a unique entry point in main Flash memory. See Section Embedded Flash memory (FLASH).

Physical remap

Once the boot pins mode is selected, the application software can modify the memory accessible in the code area (in this way the code can be executed through the ICode bus in place of the System bus). This modification is performed by programming the *Section 10.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller.
The following memories can thus be remapped:

- Main Flash memory

- System memory
- Embedded SRAM1
- FSMC bank 1 (NOR/PSRAM 1 and 2)
- QUADSPI memory

Table 6. Memory mapping versus boot mode/physical remap[1]

Addresses	Boot/remap in main Flash memory	Boot/remap in embedded SRAM 1	Boot/remap in system memory	Remap in FSMC	Remap in QUADSPI
0x2000 0000 - 0x2002 3FFF	SRAM1	SRAM1	SRAM1	SRAM1	SRAM1
0x1FFF 7000 - 0x1FFF FFFF	System memory/OTP/ Options bytes	System memory/OTP/ Options bytes	System memory/OTP/ Options bytes	System memory/OTP/ Options bytes	System memory/OTP/ Options bytes
0x1000 8000 - 0x1FFE FFFF	Reserved	Reserved	Reserved	Reserved	Reserved
0x1000 0000 - 0x1000 7FFF	CCM SRAM	CCM SRAM	CCM SRAM	CCM SRAM	CCM SRAM
0x0808 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved	Reserved	Reserved
0x0800 0000 - 0x0807 FFFF	Flash memory	Flash memory	Flash memory	Flash memory	Flash memory
0x0400 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved	FSMC bank 1 NOR/ PSRAM 2 (128 MB) Aliased	QUADSPI bank (128 MB) Aliased
0x0010 0000 - 0x03FF FFFF	Reserved	Reserved	Reserved	FSMC bank 1 NOR/ PSRAM 1 (128 MB) Aliased	QUADSPI bank (128 MB) Aliased
0x0000 0000 - 0x0007 FFFF [2] [3]	Flash Aliased	SRAM1 Aliased	System memory (28 KB) Aliased	FSMC bank 1 NOR/ PSRAM 1 (128 MB) Aliased)	QUADSPI Aliased)

1. Reserved memory area highlighted in gray in the table.
2. When the FSMC is remapped at address 0x0000 0000, only the first two regions of bank 1 memory controller (bank 1 NOR/PSRAM 1 and NOR/PSRAM 2) can be remapped. When the FSMC is remapped at address 0x0000 0000, only 128 MB are remapped. In remap mode, the CPU can access the external memory via ICode bus instead of system bus, which boosts up the performance.
3. Even when aliased in the boot memory space, the related memory is still accessible at its original memory space.

Embedded boot loader

The embedded boot loader is located in the system memory, programmed by ST during production. Refer to AN2606 STM32 microcontroller system memory boot mode.

3 Embedded Flash memory (FLASH) for category 3 devices

3.1 Introduction

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

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

3.2 FLASH main features

- Up to 512 Kbyte of Flash memory with dual bank architecture supporting read-while-write capability (RWW).
- Flash memory read operations with two data width modes supported:
  - Single bank mode DBANK=0: read access of 128 bits
  - Dual bank mode DBANK=1: read access of 64 bits
- Page erase, bank erase and mass erase (both banks)

Flash memory interface features:

- Flash memory read operations
- Flash memory program/erase operations
- Read protection activated by option (RDP)
- 4 Write protection areas (2 per bank when DBANK=1 and 4 for full memory when DBANK=0)
- 2 proprietary code read protection areas (1 per bank when DBANK=1, 2 for all memory when DBANK=0)
- 2 Securable memory areas defined by option (1 per bank when DBANK = 1, 1 for all memory when DBANK = 0).
- Prefetch on ICODE
- Instruction Cache: 32 cache lines of 4 x 64 or 2 x 128 bits on ICode (1 KB RAM)
- Data Cache: 8 cache lines of 4 x 64 bits or 2 x 128 on DCode (256B RAM)
- Error Code Correction ECC: 8 bits per 64-bit double-word
  - DBANK=1: 8 + 64 = 72 bits, 2 bits detection, 1 bit correction
  - DBANK=0: (8+64) + (8+64) = 144 bits, 2 bits detection, 1 bit correction
- Option byte loader
- Low-power mode

3.3 FLASH functional description

3.3.1 Flash memory organization

The Flash memory has the following main features:

- Capacity up to 512 Kbytes, in single bank mode (read width of 128 bits) or in dual bank mode (read width of 64-bits)
- Supports dual boot mode thanks to the BFB2 option bit (only in dual bank mode)
- Dual bank mode when DBANK bit is set:
  - 512 KB organized in 2 banks for main memory
  - Page size of 2 Kbyte
  - 72 bits wide data read (64 bits plus 8 ECC bits)
  - Bank and Mass erase
- Single bank mode when DBANK is reset:
  - 512 KB organized in one single bank for main memory
  - Page size of 4 Kbyte
  - 144 bits wide data read (128 bits plus 2x8 ECC bits)
  - Mass erase

The Flash memory is organized as follows:

- A main memory block organized depending on the dual bank configuration bit:
  - When dual bank is enabled (DBANK bit set), the Flash is divided in 2 banks of 256 KB, and each bank is organized as follows:

    The main memory block containing 128 pages of 2 Kbyte

    Each page is composed of 8 rows of 256 bytes
  - When dual bank is disabled (DBANK bit reset), the main memory block is organized as one single bank of 512 KB as follows:

    The main memory block containing 128 pages of 4 Kbyte

    Each page is composed of 8 rows of 512 bytes
- An Information block containing:
  - System memory from which the device boots in System memory boot mode. The area is reserved for use by STMicroelectronics and contains the boot loader that is used to reprogram the Flash memory through one of the following interfaces: USART, SPI, I2C, FDCAN, USB. It is programmed by STMicroelectronics when the device is manufactured, and protected against spurious write/erase operations. For further details, please refer to the AN2606 available from *www.st.com*.
  - 1 Kbyte (128 double word) OTP (one-time programmable) bytes for user data. The OTP area is available in Bank 1 only. The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double word 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 *Table 28*.

Table 7. Flash module - 512/256/128 KB dual bank organization (64 bits read width)

Flash area		Flash memory addresses	Size (bytes)	Name
Main memory (512/256/128 KB)	Bank 1 (256/128/64 KB)	0x0800 0000 - 0x0800 07FF	2 K	Page 0
		0x0800 0800 - 0x0800 0FFF	2 K	Page 1
		0x0800 1000 - 0x0800 17FF	2 K	Page 2
		0x0800 1800 - 0x0800 1FFF	2 K	Page 3
		- - - -	- - - -	- - - -
		0x0803 F800 - 0x0803 FFFF	2 K	Page 127
	Bank 2 (256/128/64 KB)	0x0804 0000 - 0x0804 07FF	2 K	Page 0
		0x0804 0800 - 0x0804 0FFF	2 K	Page 1
		0x0804 1000 - 0x0804 17FF	2 K	Page 2
		0x0804 1800 - 0x0804 1FFF	2 K	Page 3
		- - - -	- - - -	- - - -
		0x0807 F800 - 0x0807 FFFF	2 K	Page 127
Information block	Bank 1	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	Bank 2	0x1FFF 8000 - 0x1FFF EFFF	28 K	
	Bank 1	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	Bank 1	0x1FFF 7800 - 0x1FFF 782F	48	Option bytes
	Bank 2	0x1FFF F800 - 0x1FFF F82F	48	

Table 8. Flash module - 512/256/128 KB single bank organization (128 bits read width)

Flash area		Flash memory addresses	Size (bytes)	Name
Main memory (512/256/128 KB)		0x0800 0000 - 0x0800 0FFF	4 K	Page 0
		0x0800 1000 - 0x0800 1FFF	4 K	Page 1
		0x0800 2000 - 0x0800 2FFF	4 K	Page 2
		-	-	-
		-	-	-
		-	-	-
		-	-	-
		-	-	-
		-	-	-
		-	-	-
		0x0807 F000 - 0x0807 FFFF	4 K	Page 127
Information block	Bank 1	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	Bank 2	0x1FFF 8000 - 0x1FFF EFFF	28 K	
	Bank 1	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	Bank 1	0x1FFF 7800 - 0x1FFF 782F	48	Option bytes
	Bank 2	0x1FFF F800 - 0x1FFF F82F	48	

3.3.2 Error code correction (ECC)

Dual bank mode (DBANK=1, 64-bits data width)

Data in Flash memory are 72-bits words: 8 bits are added per double word (64 bits). The ECC mechanism supports:

- One error detection and correction
- Two errors detection

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

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

When an ECC error is detected, the address of the failing double word and its associated bank are saved in ADDR_ECC[20:0] and BK_ECC in the FLASH_ECCR register. ADDR_ECC[2:0] are always cleared.

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

Single bank mode (DBANK=0, 128-bits data width)

Data in Flash memory are 144-bits words: 8 bits are added per each double word. The ECC mechanism supports:

- One error detection and correction
- Two errors detection per 64 double words

The user must first check the SYSF_ECC bit, and if it is set, the user must refer to the DBANK=1 programming model (because system Flash is always on 2 banks). If the bit is not set, the user must refer to the following programing model:

Each double word (bits 63:0 and bits 127:64) has ECC.

When one error is detected in 64 LSB bits (bits 63:0) and corrected, a flag ECCC (ECC correction) is set in the FLASH_ECCR register.

When one error is detected in 64 MSB bits (bits 127:64) and corrected, a flag ECCC2 (ECC2 correction) is set in the FLASH_ECCR register.

If the ECCCIE is set, an interrupt is generated. The user has to read ECCC and ECCC2 to see which part of the 128-bits data has been corrected (either 63:0, 127:64 or both).

When two errors are detected in 64 LSB bits, a flag ECCD (ECC detection) is set in the FLASH_ECCR register.

When two errors are detected in 64 MSB bits (bits 127:64), a flag ECCD2 (ECC2 detection) is set in the FLASH_ECCR register.

In this case, a NMI is generated. The user has to read ECCD and ECCD2 to see which part of the 128-bits data has error detection (either 63:0, 127:64 or both).

When an ECC error is detected, the address of the failing the 2 times double word is saved into ADDR_ECC[20:0] in FLASH_ECCR. ADDR_ECC[20:0] contains an address of a 2 times double word.

The ADDR_ECC[3:0] are always cleared. BK_ECC is not used in this mode.

When ECCC/ECCC2 or ECCD/ECCD2 is/are set, if a new ECC error occurs, the ADDR_ECC is not updated. The FLASH_ECCR is updated only if the ECC flags (ECCC/ECCC2/ECCD/ECCD2) are cleared.

Note: For a virgin data: 0xFF FFFF FFFF FFFF FFFF, one error is detected and corrected but two errors detection 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.

3.3.3 Read access latency

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the *Flash access control register (FLASH_ACR)* according to the frequency of the CPU clock (HCLK) and the internal voltage range of the device V_{CORE}. Refer to *Section 6.1.5: Dynamic voltage scaling management*. *Table 9* shows the correspondence between wait states and CPU clock frequency.

Table 9. Number of wait states according to CPU clock (HCLK) frequency

Wait states (WS) (LATENCY)	HCLK (MHz)		
	V_{CORE} Range 1 boost mode	V_{CORE} Range 1 normal mode	V_{CORE} Range 2
0 WS (1 CPU cycles)	≤ 34	≤ 30	≤ 12
1 WS (2 CPU cycles)	≤ 68	≤ 60	≤ 24
2 WS (3 CPU cycles)	≤ 102	≤ 90	≤ 26
3 WS (4 CPU cycles)	≤ 136	≤ 120	-
4 WS (5 CPU cycles)	≤ 170	≤ 150	-

After reset, the CPU clock frequency is 16 MHz and 1 wait state (WS) is configured in the FLASH_ACR register.

When changing the CPU frequency, the following software sequences must be applied in order to tune the number of wait states needed to access the Flash memory:

Increasing the CPU frequency:

1. Program the new number of wait states to the LATENCY bits in the *Flash access control register (FLASH_ACR)*.
2. Check that the new number of wait states is taken into account to access the Flash memory by reading the FLASH_ACR register.
3. Analyze the change of CPU frequency change caused either by:
   - changing clock source defined by SW bits in RCC_CFGR register
   - or by CPU clock prescaller defined by HPRE bits in RCC_CFGR

If some of above two steps decreases the CPU frequency, firstly perform this step and then the rest. Otherwise modify The CPU clock source by writing the SW bits in the RCC_CFGR register and then (if needed) modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.

4. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register.

Decreasing the CPU frequency:

1. Modify the CPU clock source by writing the SW bits in the RCC_CFGR register.
2. If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.
3. Analyze the change of CPU frequency change caused either by:
   - changing clock source defined by SW bits in RCC_CFGR register
   - or by CPU clock prescaller defined by HPRE bits in RCC_CFGR

If some of above two steps increases the CPU frequency, firstly perform another step and then this step. Otherwise modify The CPU clock source by writing the SW bits in the

RCC_CFGR register and then (if needed) modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.

4. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register.
5. Program the new number of wait states to the LATENCY bits in *Flash access control register (FLASH_ACR)*.
6. Check that the new number of wait states is used to access the Flash memory by reading the FLASH_ACR register.

3.3.4 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 FPU processors. It balances the inherent performance advantage of the Arm® Cortex®-M4 with FPU over Flash memory technologies, which normally requires 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 which 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 Flash memory at a CPU frequency up to 170 MHz.

Instruction prefetch

The Cortex®-M4 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.

In case of Single bank mode (DBANK option bit is reset), each Flash memory read operation provides 128 bits from either four instructions of 32 bits or eight instructions of 16 bits depending on the launched program. This 128-bits current instruction line is saved in a current buffer, and in case of sequential code, at least four CPU cycles are needed to execute the previous read instruction line.

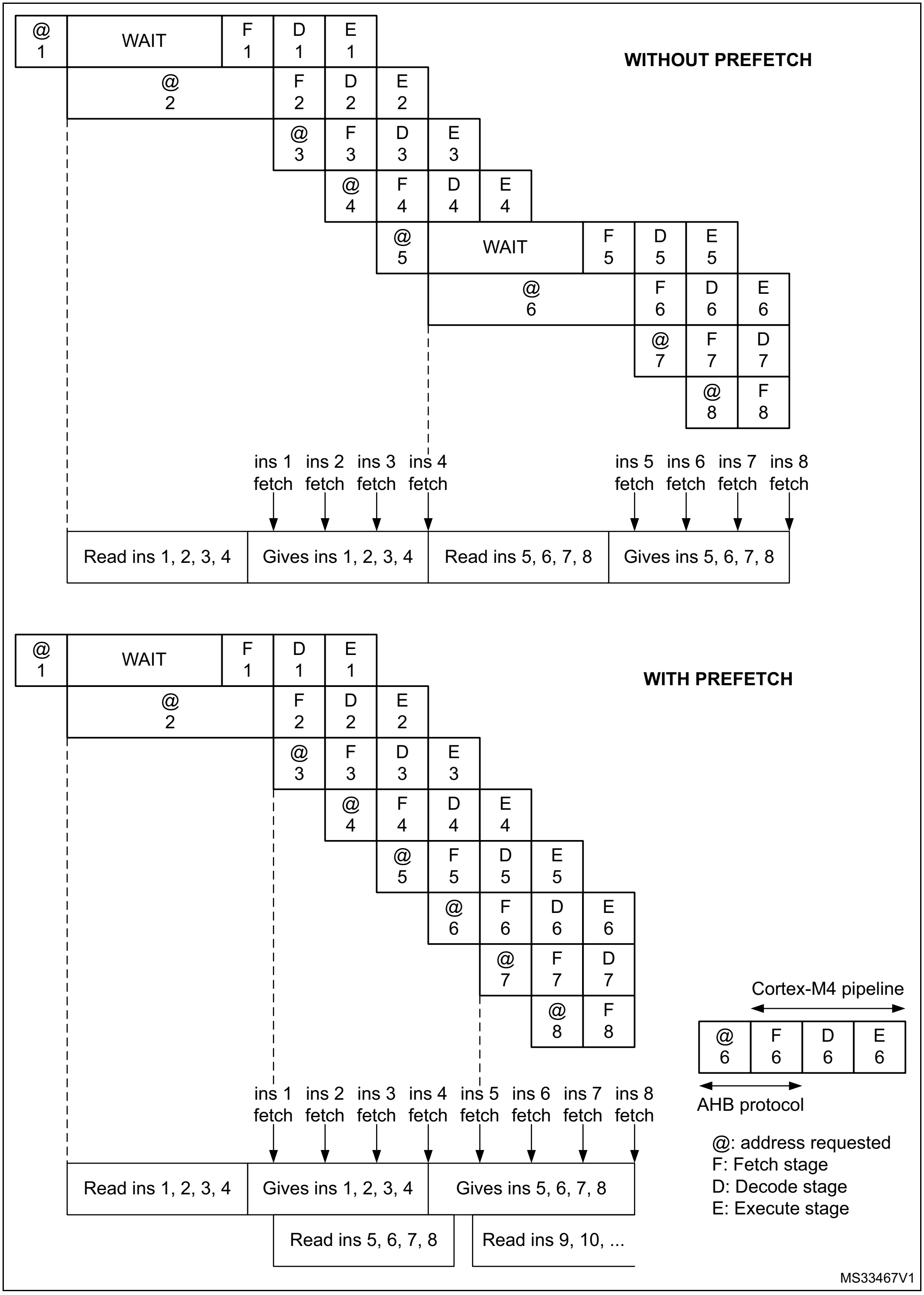
When in dual bank mode (DBANK option bit is set), each Flash memory read operation provides 64 bits from either two instructions of 32 bits or four instructions of 16 bits depending on the launched program. This 64-bits current instruction line is saved in a current buffer, and in case of sequential code, at least two CPU cycles are needed to execute the previous read instruction line.

Prefetch on the 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 the PRFTEN bit in the *Flash access control register (FLASH_ACR)*. This feature is useful if at least one wait state is needed to access the Flash memory.

Figure 9 shows the execution of sequential 16-bit instructions with and without prefetch when 3 WS are needed to access the Flash memory.

Figure 3. Sequential 16-bit instructions execution (64-bit read data width)



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 flash access is performed.

Instruction cache memory (I-Cache)

To limit the time lost due to jumps, it is possible to retain 32 lines of 4 x 64 bits in dual bank mode or 32 lines of 2 x 128 bits in single bank mode in an instruction cache memory.This feature can be enabled by setting the instruction cache enable (ICEN) bit in the *Flash access control register (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 enable after system reset.

Data cache memory (D-Cache)

Literal pools are fetched from Flash memory through the DCode bus during the execution stage of the CPU pipeline. Each DCode bus read access fetches 64 or 128 bits which 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 databus DCode have priority over accesses through the AHB instruction bus ICode.

If some literal pools are frequently used, the data cache memory can be enabled by setting the data cache enable (DCEN) bit in the *Flash access control register (FLASH_ACR)*. This feature works like the instruction cache memory, but the retained data size is limited to 8 rows of 4*64 bits in dual bank mode and to 8 rows of 2*128 bits in single bank mode.

The Data cache memory is enable after system reset.

Note: The D-Cache is active only when data is requested by the CPU (not by DMA1 and DMA2). Data in option bytes block are not cacheable.

3.3.5 Flash program and erase operations

The STM32G4 Series 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 boot loader to load the user application 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 (I/Os, USB, CAN, UART, I^{2}C, SPI, etc.) 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 has to 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.

An on-going Flash memory operation does not block the CPU as long as the CPU does not access the same Flash memory bank. Code or data fetches are possible on one bank while a write/erase operation is performed to the other bank (refer to *Section 3.3.8: Read-while-write (RWW) available only in dual bank mode (DBANK=1)*).

The Flash erase and programming is only possible in the voltage scaling range 1. The VOS[1:0] bits in the PWR_CR1 must be programmed to 01b.

On the contrary, during a program/erase operation to the Flash memory, any attempt to read the same Flash memory bank stalls the bus. The read operation proceeds correctly once the program/erase operation has completed.

Unlocking the Flash memory

After reset, write is not allowed in the *Flash control register (FLASH_CR)* to protect the Flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence is used to unlock this register:

1. Write KEY1 = 0x45670123 in the *Flash key register (FLASH_KEYR)*
2. Write KEY2 = 0xCDEF89AB in the FLASH_KEYR register.

Any wrong sequence locks up the FLASH_CR 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 FLASH_CR register can be locked again by software by setting the LOCK bit in the FLASH_CR register.

Note: The FLASH_CR register cannot be written when the BSY bit in the Flash status register (FLASH_SR) is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

3.3.6 Flash main memory erase sequences

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

Page erase

To erase a page, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH_SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. In dual bank mode (DBANK option bit is set), set the PER bit and select the page to erase (PNB) with the associated bank (BKER) in the Flash control register (FLASH_CR). In single bank mode (DBANK option bit is reset), set the PER bit and select the page to erase (PNB). The BKER bit in the Flash control register (FLASH_CR) must be kept cleared.
4. Set the STRT bit in the FLASH_CR register.
5. Wait for the BSY bit to be cleared in the FLASH_SR register.

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

If the page erase is part of write-protected area (by WRP or PCROP), WRPERR is set and the page erase request is aborted.

Bank 1, Bank 2 Mass erase (available only in dual bank mode when DBANK=1)

To perform a bank Mass Erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the MER1 bit or MER2 (depending on the bank) in the *Flash control register (FLASH_CR)*. Both banks can be selected in the same operation, in that case it corresponds to a mass erase.
4. Set the STRT bit in the FLACH_CR register.
5. Wait for the BSY bit to be cleared in the *Flash status register (FLASH_SR)*.

Mass erase

To perform a Mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the MER1 bit and MER2 in the Flash control register (FLASH_CR).
4. Set the STRT bit in the FLACH_CR register.
5. Wait for the BSY bit to be cleared in the Flash status register (FLASH_SR).

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

When DBANK=0, if only the MERA or the MERB bit is set, PGSERR is set and no erase operation is performed.

If the bank to erase or if one of the banks to erase contains a write-protected area (by WRP or PCROP), WRPERR is set and the mass erase request is aborted (for both banks if both are selected).

3.3.7 Flash main memory programming sequences

The Flash memory is programmed 72 bits at a time (64 bits + 8 bits ECC).

Programming in a previously programmed address is not allowed except if the data to write is full zero, and any attempt sets PROGERR flag in the *Flash status register (FLASH_SR).*

It is only possible to program double word (2 x 32-bit data).

- Any attempt to write byte or half-word sets SIZERR flag in the FLASH_SR register.
- Any attempt to write a double word which is not aligned with a double word address sets PGAERR flag in the FLASH_SR register.

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 the BSY bit in the *Flash status register (FLASH_SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PG bit in the *Flash control register (FLASH_CR)*.
4. Perform the data write operation at the desired memory address, inside main memory block or OTP area. Only double word can be programmed.
   – Write a first word in an address aligned with double word
   – Write the second word
5. Wait until the BSY bit is cleared in the FLASH_SR register.
6. Check that EOP flag is set in the FLASH_SR register (meaning that the programming operation has succeed), and clear it by software.
7. Clear the PG bit in the FLASH_SR register if there no more programming request anymore.

Note: When the flash interface has received a good sequence (a double word), programming is automatically launched and BSY bit is set. The internal oscillator HSI16 (16 MHz) is enabled automatically when PG bit is set, and disabled automatically when PG bit is cleared, except if the HSI16 is previously enabled with HSION in 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.

Fast programming for a row (64 double words if DBANK=1) or for half row (64 double words if DBANK=0)

This mode allows to program a row (64 double words if DBANK=1) or half row (64 double words if DBANK=0), and to reduce the page programming time by eliminating the need for verifying the flash locations before they are programmed and to avoid rising and falling time of high voltage for each double word. During fast programming, the CPU clock frequency (HCLK) must be at least 8 MHz.

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

The Flash main memory programming sequence in standard mode is as follows:

1. In single bank mode (DBANK=0), perform a mass erase. If not, PGSERR is set. The Fast programing can be performed only if the code is executed from RAM or from bootloader. In dual bank mode (DBANK=1), perform a mass erase of the bank to program. If not, PGSERR is set.
2. Check that no Flash main memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH_SR)*.
3. Check and clear all error programming flag due to a previous programming.

4. Set the FSTPG bit in *Flash control register (FLASH_CR)*.
5. Write the 64 double words to program a row or half row. Only double words can be programmed:
   - Write a first word in an address aligned with double word
   - Write the second word.
6. Wait until the BSY bit is cleared in the FLASH_SR register.
7. Check that EOP flag is set in the FLASH_SR register (meaning that the programming operation has succeed), and clear it by software.
8. Clear the FSTPG bit in the FLASH_SR register if there no more programming request anymore.

Note: If the flash is attempted to be written in Fast programming mode while a read operation is on going in the same bank, the programming is aborted without any system notification (no error flag is set).

When the Flash interface has received 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 has been programmed or in case of error. The internal oscillator HSI16 (16 MHz) is enabled automatically when FSTPG bit is set, and disabled automatically when FSTPG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

The 64 double word must be written successively. The high voltage is kept on the flash for all the programming. Maximum time between two double words write requests is the time programming (around 2 x 25us). If a second double word arrives after this time programming, fast programming is interrupted and MISSERR is set.

High voltage mustn't exceed 8 ms for a full row between 2 erases. This is guaranteed by the sequence of 64 double words successively written with a clock system greater or equal to 8MHz. An internal time-out counter counts 7ms when Fast programming is set and stops the programming when time-out is over. In this case the FASTERR bit 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

Several kind of errors are 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 is not previously erased (except if the value to program is full zero).
- **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 doesn’t belong to the same double word address.
  - In fast programming: the data to program doesn’t 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: MER1, MER2, and PER are not cleared when PG or FSTPG is set.
  - In the fast programming sequence: the Mass erase is not performed before setting FSTPG bit.
  - In the mass erase sequence: PG, FSTPG, and PER are not cleared when MER1 or MER2 is set.
  - In the page erase sequence: PG, FSTPG, MER1 and MER2 are not cleared when PER is set.
  - PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR or PGSERR is set due to a previous programming error.
  - When DBANK=0, in the case that only either MER1 or MER2 is set, PGSERR is set (bank mass erase is not allowed).
- **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 or in a Securable memory area.
  - Attempt to perform a bank erase when one page or more is protected by WRP or PCROP.
  - The debug features are connected or the boot is executed from SRAM or from System flash when the read protection (RDP) is set to Level 1.
  - Attempt to modify the option bytes when the read 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 programmation 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 7ms which generates a time-out detection.

  – When the fast programming has been 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 the FLASH_SR register:

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 the *Flash status register (FLASH_SR)*, an interrupt is generated and the operation error flag OPERR is set in the FLASH_SR register.

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 requests.

Programming and caches

If a Flash memory write access concerns some data in the data cache, the Flash write access modifies the data in the Flash memory and the data in the cache.

If an erase operation in Flash memory also concerns data in the data or instruction cache, you have to make sure that these data are rewritten before they are accessed during code execution. If this cannot be done safely, it is recommended to flush the caches by setting the DCRST and ICRST bits in the *Flash access control register (FLASH_ACR)*.

Note: The I/D cache should be flushed only when it is disabled (I/DCEN = 0).

3.3.8 Read-while-write (RWW) available only in dual bank mode (DBANK=1)

The dual bank mode is available only when the DBANK option bit is set, allowing read-while-write operations. This feature allows to perform a read operation from one bank while an erase or program operation is performed to the other bank.

Note: Write-while-write operations are not allowed. As an example, It is not possible to perform an erase operation on one bank while programming the other one.

Read from bank 1 while page erasing in bank 2 (or vice versa)

While executing a program code from bank 1, it is possible to perform a page erase operation on bank 2 (and vice versa). Follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH_SR)* (BSY is active when erase/program operation is on going in bank 1 or bank 2).
2. Set PER bit, PSB to select the page and BKER to select the bank in the *Flash control register (FLASH_CR)*.
3. Set the STRT bit in the FLASH_CR register.
4. Wait for the BSY bit to be cleared (or use the EOP interrupt).

Read from bank 1 while mass erasing bank 2 (or vice versa)

While executing a program code from bank 1, it is possible to perform a mass erase operation on bank 2 (and vice versa). Follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH_SR)* (BSY is active when erase/program operation is on going in bank 1 or bank 2).
2. Set MER1 or MER2 to in the *Flash control register (FLASH_CR)*.
3. Set the STRT bit in the FLASH_CR register.
4. Wait for the BSY bit to be cleared (or use the EOP interrupt).

Read from bank 1 while programming bank 2 (or vice versa)

While executing a program code from bank 1, it is possible to perform a program operation on the bank 2. (and vice versa). Follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH_SR)* (BSY is active when erase/program operation is on going on bank 1 or bank 2).
2. Set the PG bit in the *Flash control register (FLASH_CR)*.
3. Perform the data write operations at the desired address memory inside the main memory block or OTP area.
4. Wait for the BSY bit to be cleared (or use the EOP interrupt).

3.4 FLASH option bytes

3.4.1 Option bytes description

The option bytes are configured by the end user depending on the application requirements. As a configuration example, the watchdog may be selected in hardware or software mode (refer to *Section 5.4.2: Option bytes programming*).

A double word is split up as follows in the option bytes:

Table 10. Option byte format

63-24	23-16	15 -8	7-0	31-24	23-16	15 -8	7-0
Complemented option byte 3	Complemented option byte 2	Complemented option byte 1	Complemented option byte 0	Option byte 3	Option byte 2	Option byte 1	Option byte 0

The organization of these bytes inside the information block is as shown in *Table 31: Option byte organization*.

The option bytes can be read from the memory locations listed in *Table 31: Option byte organization* or from the Option byte registers:

- *Flash option register (FLASH_OPTR)*
- *Flash PCROP1 Start address register (FLASH_PCROP1SR)*
- *Flash PCROP1 End address register (FLASH_PCROP1ER)*
- *Flash WRP area A address register (FLASH_WRP1AR)*
- *Flash WRP area B address register (FLASH_WRP1BR)*
- *Flash PCROP2 Start address register (FLASH_PCROP2SR)*
- *Flash PCROP2 End address register (FLASH_PCROP2ER)*
- *Flash Bank 2 WRP area A address register (FLASH_WRP2AR)*
- *Flash Bank 2 WRP area B address register (FLASH_WRP2BR)*.

Table 11. Option byte organization

BANK	Address	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
Bank 1	1FFF7800	$\overline{\text{USER OPT}}$			$\overline{\text{RDP}}$	USER OPT			RDP
	1FFF7808	$\overline{\text{Unused}}$		$\overline{\text{Unused and}}$ $\overline{\text{PCROP1\_STRT[14:0]}}$		Unused		Unused and PCROP1_STRT[14:0]	
	1FFF7810	$\overline{\text{PCROP\_RDP and}}$ $\overline{\text{Unused}}$		$\overline{\text{Unused and}}$ $\overline{\text{PCROP1\_END[14:0]}}$		PCROP_RDP and Unused		Unused and PCROP1_END[14:0]	
	1FFF7818	$\overline{\text{Unused}}$	$\overline{\text{WRP1A\_END}}$ [6:0]	$\overline{\text{Unused}}$	$\overline{\text{WRP1A\_STRT}}$ [6:0]	Unused	WRP1A_END [6:0]	Unused	WRP1A_STRT [6:0]
	1FFF7820	$\overline{\text{Unused}}$	$\overline{\text{WRP2A\_END}}$ [6:0]	$\overline{\text{Unused}}$	$\overline{\text{WRP2A\_STRT}}$ [6:0]	Unused	WRP2A_END [6:0]	Unused	WRP2A_STRT [6:0]
	1FFF7828	$\overline{\text{Unused}}$	$\overline{\text{BOOT\_LOCK}}$	$\overline{\text{Unused}}$	$\overline{\text{SEC\_SIZE1}}$	Unused	BOOT_LOCK	Unused	SEC_SIZE1

Table 11. Option byte organization (continued)

BANK	Address	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
Bank 2	1FFFF800	Unused							
	1FFFF808	$\overline{\text{Unused}}$		$\overline{\text{Unused and}}$ $\overline{\text{PCROP2\_STRT[14:0]}}$		Unused		Unused and PCROP2_STRT[14:0]	
	1FFFF810	$\overline{\text{Unused}}$		$\overline{\text{Unused and}}$ $\overline{\text{PCROP2\_END[14:0]}}$		Unused		Unused and PCROP2_END[14:0]	
	1FFFF818	$\overline{\text{Unused}}$	$\overline{\text{WRP2B\_END[6:0]}}$	$\overline{\text{Unused}}$	$\overline{\text{WRP2B\_STRT[6:0]}}$	Unused	WRP2B_END[6:0]	Unused	WRP2B_STRT[6:0]
	1FFFF820	$\overline{\text{Unused}}$	$\overline{\text{WRP2B\_END[6:0]}}$	$\overline{\text{Unused}}$	$\overline{\text{WRP2B\_STRT[6:0]}}$	Unused	WRP2B_END[6:0]	Unused	WRP2B_STRT[6:0]
	1FFFF828	$\overline{\text{Unused}}$			$\overline{\text{SEC\_SIZE2}}$	Unused			SEC_SIZE2

User and read protection option bytes

Flash memory address: 0x1FFF 7800

ST production value: 0xFFEF F8AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IRH_EN	NRST_MODE		nBOOT0	nSWBOOT0	CCMSRAM_RST	SRAM_PE	nBOOT1	DBANK	Res.	BFB2	WWDG_SW	IWGD_STDBY	IWDG_STOP	IWDG_SW
	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.	nRST_SHDW	nRST_STDBY	nRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
	r	r	r		r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **IRH_IN**: Internal reset holder for PG10
0: IRH disabled
1: IRH enabled

Bits 29:28 **NRST_MODE**: PG10 pad mode
00: Reset Input/Output
01: Reset Input only
10: GPIO
11: Reset Input/Output

Bit 27 **nBOOT0:** nBOOT0 option bit
0: nBOOT0 = 0
1: nBOOT0 = 1

Bit 26 **nSWBOOT0:** Software BOOT0
0: BOOT0 taken from the option bit nBOOT0
1: BOOT0 taken from PB8/BOOT0 pin

Bit 25 **CCMSRAM_RST:** CCM SRAM erase when system reset
0: CCM SRAM erased when a system reset occurs
1: CCM SRAM is not erased when a system reset occurs

Bit 24 **SRAM_PE:** SRAM1 and CCM SRAM parity check enable
0: SRAM1 and CCM SRAM parity check enable
1: SRAM1 and CCM SRAM parity check disable

Bit 23 **nBOOT1:** Boot configuration
Together with the BOOT0 pin, this bit selects boot mode from the Flash main memory, SRAM1 or the System memory. Refer to *Section 2.6: Boot configuration*.

Bit 22 **DBANK:**
0: Single bank mode with 128 bits data read width
1: Dual bank mode with 64 bits data
This bit can be written only when PCROP1/2 is disabled.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **BFB2:** Dual-bank boot
0: Dual-bank boot disable
1: Dual-bank boot enable

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 is frozen in Standby mode
1: Independent watchdog counter is running in Standby mode

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

Bit 16 **IDWG_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**
0: Reset generated when entering the Shutdown mode
1: No reset generated when entering the Shutdown mode

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

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

Bit 11 Reserved, must be kept at reset value.

Bits10:8 **BOR_LEV:** BOR reset Level

These bits contain the VDD 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

Bits 7:0 **RDP:** Read protection level

0xAA: Level 0, read protection not active
0xCC: Level 2, chip read protection active
Others: Level 1, memories read protection active

PCROP1 Start address option bytes

Flash memory address: 0x1FFF 7808

ST production value: 0xFFFF FFFF

31	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.	PCROP1_STRT[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PCROP1_STRT:** PCROP area start offset

DBANK=1
PCROP1_STRT contains the first double-word of the PCROP area for bank1.
DBANK=0
PCROP1_STRT contains the first 2xdouble-word of the PCROP area for all memory.

PCROP1 End address option bytes

Flash memory address: 0x1FFF 7810

ST production value: 0x00FF 0000

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.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PCROP1_END[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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 is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **PCROP1_END:** Bank 1 PCROP area end offset

DBANK=1

PCROP1_END contains the last double-word of the bank 1 PCROP area.

DBANK=0

PCROP1_END contains the last 2x double-word PCROP area for all memory.

WRP1 Area A address option bytes

Flash memory address: 0x1FFF 7818

ST production value: 0xFF00 FFFF

31	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]						
									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.	Res.	WRP1A_STRT[6:0]						
									r	r	r	r	r	r	r

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

Bits 22:16 **WRP1A_END:** WRP first area “A” end offset

DBANK=1

WRP1A_END contains the last page of WRP first area in bank1.

DBANK=0

WRP1A_END contains the last page of WRP first area for all memory.

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

Bits 6:0 **WRP1A_STRT:** WRP first area “A” start offset

DBANK=1

WRP1A_STRT contains the first page of WRP first area for bank1.

DBANK=0

WRP1A_STRT contains the first page of WRP first area for all memory.

WRP2 Area A address option bytes

Flash memory address: 0x1FFF 7820

ST production value: 0xFF00 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_END[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_STRT[6:0]						
									r	r	r	r	r	r	r

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

Bits 22:16 **WRP2A_END:** WRP second area “B” end offset

DBANK=1
WRP2A_END contains the last page of the WRP second area for bank1.
DBANK=0
WRP2A_END contains the last page of the WPR second area for all memory.

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

Bits 6:0 **WRP2A_STRT:** WRP second area start offset

DBANK=1
WRP2A_STRT contains the last page of the WRP second area for bank1.
DBANK=0
WRP2A_STRT contains the last page of the WPR second area for all memory.

Securable memory area Bank 1 option bytes

Flash memory address: 0x1FFF7828

ST production value: 0xFF00 FF00

31	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.	BOOT_LOCK
															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.	SEC_SIZE1[7:0]							
								r	r	r	r	r	r	r	r

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

Bit 16 **BOOT_LOCK** used to force boot from user area

0: Boot based on the pad/option bit configuration
1: Boot forced from Main Flash memory

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SEC_SIZE1[7:0]:** Securable memory area size

Contains the number of Securable Flash memory pages

PCROP2 Start address option bytes

Flash memory address: 0x1FFFF808

ST production value: 0xFFFF FFFF

31	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.	PCROP2_STRT[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PCROP2_STRT:** PCROP area start offset
DBANK=1
PCROP2_STRT contains the first double-word of the PCROP area for bank 2.
DBANK=0
PCROP2_STRT contains the first double-word PCROP area for all memory.

PCROP2 End address option bytes

Flash memory address: 0x1FFF F810

ST production value: 0x00FF 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.	PCROP2_END[14:0]														
	r	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 **PCROP2_END:** PCROP area end offset
DBANK=1
PCROP2_END contains the last double-word of the PCROP area for bank2.
DBANK=0
PCROP2_END contains the last 2xdouble-word of the PCROP area for all the memory.

WRP1 Area B address option bytes

Flash memory address: 0x1FFF F818

ST production value: 0xFF00 FFFF

31	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]						
									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.	Res.	WRP1B_STRT[6:0]						
									r	r	r	r	r	r	r

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

Bits 22:16 **WRP1B_END:** WRP first area “B” end offset

DBANK=1
WRP1B_END contains the last page of the WRP first area for bank2.

DBANK=0
WRP1B_END contains the last page of the WRP third area for all memory.

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

Bits 6:0 **WRP1B_STRT:** WRP first area “B” start offset

DBANK=1
WRP1B_STRT contains the first page of the WRP first area for bank2.

DBANK=0
WRP1B_STRT contains the first page of the WRP third area for all memory.

WRP2 Area B address option bytes

Flash memory address: 0x1FFF F820

ST production value: 0xFF00 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_END[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_STRT[6:0]						
									r	r	r	r	r	r	r

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

Bits 22:16 **WRP2B_END:** WRP second area “B” end offset

DBANK=1
WRP2B_END contains the last page of the WRP second area for bank2.
DBANK=0
WRP2B_END contains the last page of the WRP fourth area for all memory.

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

Bits 6:0 **WRP2B_STRT:** WRP second area “B” start offset

DBANK=1
WRP2B_STRT contains the first page of the WRP second area for bank2.
DBANK=0
WRP2B_STRT contains the first page of the WRP second area for all memory.

Securable memory area Bank 2 option bytes

Flash memory address: 0x1FFF F828

ST production value: 0xFF00 FF00

31	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.	SEC_SIZE2[7:0]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **SEC_SIZE2[7:0]:** Securable memory area size contains the number of Securable Flash memory pages.

3.4.2 Option bytes programming

After reset, the options related bits in the *Flash control register (FLASH_CR)* are write-protected. To run any operation on the option bytes page, the option lock bit OPTLOCK in the *Flash control register (FLASH_CR)* must be cleared. The following sequence is used to unlock this register:

1. Unlock the FLASH_CR with the LOCK clearing sequence (refer to *Unlocking the Flash memory*).
2. Write OPTKEY1 = 0x08192A3B in the *Flash option key register (FLASH_OPTKEYR)*.
3. Write OPTKEY2 = 0x4C5D6E7F in the FLASH_OPTKEYR register.

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 too.

Modifying user options

The option bytes are programmed differently from a main memory user address. It is not possible to modify independently user options of bank 1 or bank 2. The users Options of the bank 1 are modified first.

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

1. Check that no Flash memory operation is on going by checking the BSY bit in the *Flash status register (FLASH_SR)*.
2. Clear OPTLOCK option lock bit with the clearing sequence described above.
3. Write the desired options value in the options registers: *Flash option register (FLASH_OPTR)*, *Flash PCROP1 Start address register (FLASH_PCROP1SR)*, *Flash PCROP1 End address register (FLASH_PCROP1ER)*, *Flash WRP area A address register (FLASH_WRP1AR)*, *Flash WRP area B address register (FLASH_WRP1BR)*, *Flash PCROP2 Start address register (FLASH_PCROP2SR)*, *Flash PCROP2 End address register (FLASH_PCROP2ER)*, *Flash Bank 2 WRP area A address register (FLASH_WRP2AR)*, *Flash Bank 2 WRP area B address register (FLASH_WRP2BR)*.
4. Set the Options Start bit OPTSTRT in the *Flash control register (FLASH_CR)*.
5. Wait for the BSY bit to be cleared.

Note: Any modification of the value of one option is automatically performed by erasing both user option bytes pages first (bank 1 and bank 2) and then programming all the option bytes with the values contained in the flash option registers.

Option byte loading

After the BSY bit is cleared, all new options are updated into the Flash but they are not applied to the system. They have effect on the system when they are loaded. Option bytes loading (OBL) is performed in two cases:

- when OBL_LAUNCH bit is set in the *Flash control register (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 cannot be read with 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 to check the loading has correctly taken place.

During option byte loading, the options are read by double word with ECC. 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:

- For USR OPT option, the value of mismatch is all options at ‘1’, except for BOR_LEV which 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”

On system reset rising, internal option registers are copied into option registers which can be read and written by software (FLASH_OPTR, FLASH_PCROP1/2SR, FLASH_PCROP1/2ER, FLASH_WRP1/2AR, FLASH_WRP1/2BR). These registers are also used to modify options. If these registers are not modified by user, they reflects the options states of the system. See *Section : Modifying user options* for more details.

Activating dual bank mode (switching from DBANK=0 to DBANK=1)

When switching from one Flash mode to another (for example from single to dual bank) it is recommended to execute the code from the SRAM or use the bootloader. To avoid reading corrupted data from the Flash when the memory organization is changed, any access (either CPU or DMAs) to Flash memory should be avoided before reprogramming.

- Disable Instruction/data caches and/or prefetch if they are enabled (reset PRFTEN and ICEN/DCEN bits in the FLASH_ACR register).
- Flush instruction and data cache by setting the DCRST/ICRST bits in the FLASH_ACR register.
- Set the DBANK option bit and clear all the WRP write protection (follow user option modification and option bytes loader procedure).
  - Once OBL is done with DBANK=1, perform a mass erase.
  - Start a new programing of code in 64 bits mode with DBANK=1 memory mapping
  - Set the new WRP/PCROP with DBANK=1 scheme if needed.
  - Set PRFTEN and ICEN/DCEN if needed.

The new software is ready to be run using the bank configuration.

De-activating dual bank mode (switching from DBANK=1 to DBANK=0)

When switching from one Flash mode to another (for example from single to dual bank) it is recommended to execute the code from the SRAM or use the bootloader. To avoid reading corrupted data from the Flash when the memory organization is changed, any access (either CPU or DMAs) to Flash memory should be avoided before reprogramming.

- Disable Instruction/data caches and/or prefetch if they are enabled (reset PRFTEN and ICEN/DCEN bits in the FLASH_ACR register).
- Flush instruction and data cache by setting the DCRST/ICRST bits in the FLASH_ACR register.
- Clear the DBANK option bit and all WRP write protection (follow user option modification and option bytes loader procedure).
  - Once OBL is done with DBANK=0, perform a mass erase.
  - Start a new programing of code in 128 bits mode with DBANK=0 memory mapping
  - Set the new WRP/PCROP with DBANK=0 scheme if needed. Set PRFTEN and ICEN/DCEN if needed.

The new software is ready to be run using the bank configuration.

3.5 FLASH memory protection

The Flash main memory can be protected against external accesses with the Read protection (RDP). The pages of the Flash memory can also be protected against unwanted write due to loss of program counter contexts. The write-protection (WRP) granularity is one page (2 KByte). Apart of the flash memory can also be protected against read and write from third parties (PCROP). The PCROP granularity is double word (64-bit).

3.5.1 Read protection (RDP)

The read protection is activated by setting the RDP option byte and then, by applying a system reset to reload the new RDP option byte. The read protection protects to the Flash main memory, the option bytes, the backup registers (RTC_BKPxR in the RTC) and the CCM SRAM.

Note: If the read protection is set while the debugger is still connected through JTAG/SWD, apply a POR (power-on reset) instead of a system reset.

There are three levels of read 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 *Table 32*.

Table 12. Flash memory read protection status

RDP byte value	RDP complement value	Read protection level
0xAA	0x55	Level 0 (production value)
Any value except 0xAA or 0xCC	Any value (not necessarily complementary) except 0x55 and 0x33	Level 1
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 Flash main memory area are possible. The option bytes, the CCM SRAM and the backup registers are also accessible by all operations.

Level 1: Read protection

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

- **User mode:** Code executing in user mode (**Boot Flash**) can access Flash main memory, option bytes, CCM SRAM and backup registers with all operations.
- **Debug, boot RAM and boot loader modes:** In debug mode or when code is running from boot RAM or boot loader, the Flash main memory, the backup registers (RTC_BKPxR in the RTC) and the CCM SRAM are totally inaccessible. In these modes, a read or write access to the Flash generates a bus error and a Hard Fault interrupt.

Caution: In case the Level 1 is configured and no PCROP area is 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 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 Cortex®-M4 debug port, the boot from RAM (boot RAM mode) and the boot from System memory (boot loader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the Flash Main memory. On the contrary, only read operations can be performed on the option bytes.

Option bytes cannot be programmed nor erased. Thus, the level 2 cannot be removed at all: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag WRPERR is set in the Flash_SR register and an interrupt can be generated.

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.

Changing the Read 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 either directly from level 0 or from level 1. Once in level 2, it is no more possible to modify the Read protection level.

When the RDP is reprogrammed to the value 0xAA to move from Level 1 to Level 0, a mass erase of the Flash main memory is performed if PCROP_RDP is set in the *Flash PCROP1 End address register (FLASH_PCROP1ER)*. The backup registers (RTC_BKPxR in the RTC) and the CCM SRAM are also erased. The user options except PCROP protection are set to their previous values copied from FLASH_OPTR, FLASH_WRPxyR (x=1, 2 and y =A or B). PCROP is disable. The OTP area is not affected by mass erase and remains unchanged.

If the bit PCROP_RDP is cleared in the FLASH_PCROP1ER, the full mass erase is replaced by a partial mass erase that is successive page erases in the bank where PCROP is active, except for the pages protected by PCROP. This is done in order to keep the PCROP code. If PCROP is active for both banks, both banks are erased by page erases.

Only when both banks are erased, options are re-programmed with their previous values. This is also true for FLASH_PCROPxSR and FLASH_PCROPxER registers (x=1,2).

Note: Full mass erase or partial mass erase is performed only when Level 1 is active and Level 0 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 control register.

Figure 4. Changing the read protection (RDP) level

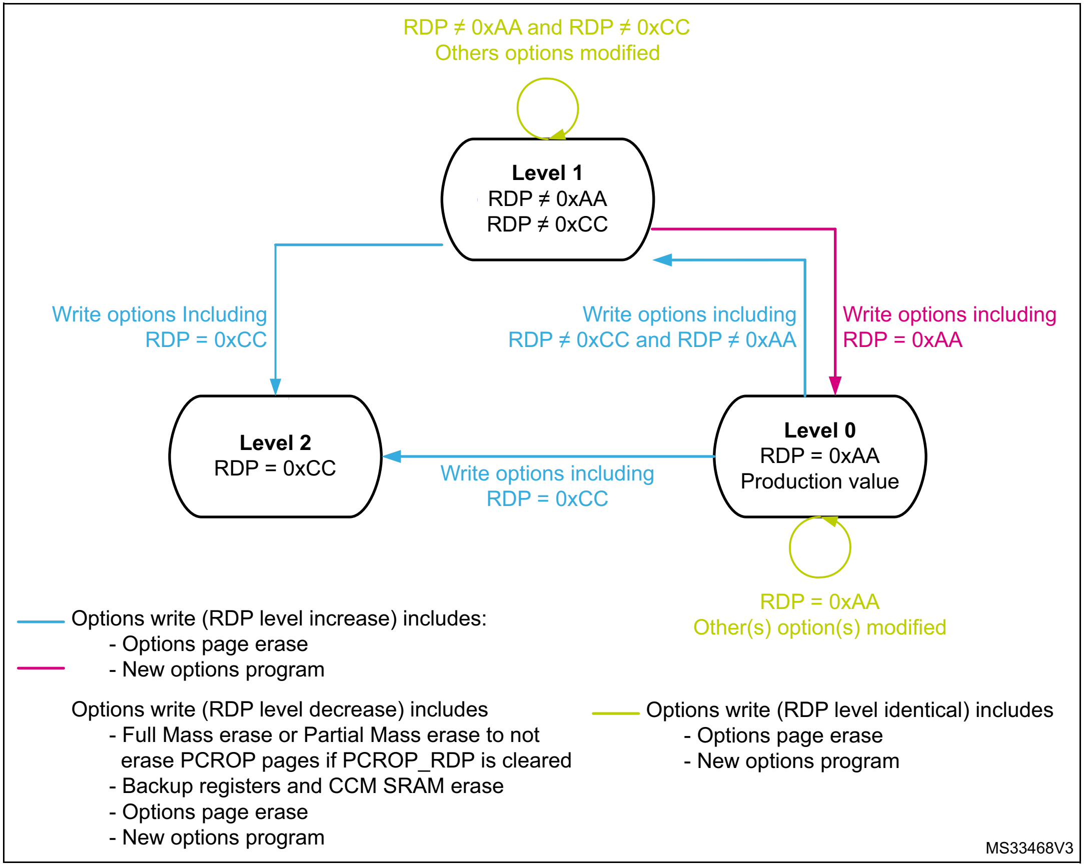


Table 13. Access status versus protection level and execution modes

Area	Protection level	User execution (BootFromFlash)			Debug/ BootFromRam/ BootFromLoader[(1)]		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No[(3)]
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory [(2)]	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes	1	Yes	Yes[(3)]	Yes	Yes	Yes[(3)]	Yes
	2	Yes	No	No	N/A	N/A	N/A

Table 13. Access status versus protection level and execution modes (continued)

Area	Protection level	User execution (BootFromFlash)			Debug/ BootFromRam/ BootFromLoader[1]		
		Read	Write	Erase	Read	Write	Erase
OTP	1	Yes	Yes[4]	N/A	No	No	N/A
	2	Yes	Yes[4]	N/A	N/A	N/A	N/A
Backup registers	1	Yes	Yes	N/A	No	No	No[5]
	2	Yes	Yes	N/A	N/A	N/A	N/A
CCM SRAM	1	Yes	Yes	N/A	No	No	No[6]
	2	Yes	Yes	N/A	N/A	N/A	N/A

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 main memory is erased when the RDP option byte is programmed with all level protections disabled (0xAA).
4. OTP can only be written once.
5. The backup registers are erased when RDP changes from level 1 to level 0.
6. The CCM SRAM is erased when RDP changes from level 1 to level 0.

3.5.2 Proprietary code readout protection (PCROP)

Apart 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, as an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. Depending of the DBANK mode, it allows either to specify one PCROP zone per bank in dual bank mode or to specify two PCROP zones for all memory. An additional option bit (PCROP_RDP) allows to select if the PCROP area is erased or not when the RDP protection is changed from Level 1 to Level 0 (refer to *Changing the Read protection level*).

Each PCROP area is defined by a start page offset and an end page offset related to the physical Flash bank base address. These offsets are defined in the PCROP address registers *Flash PCROP1 Start address register (FLASH_PCROP1SR)*, *Flash PCROP1 End address register (FLASH_PCROP1ER)*, *Flash PCROP2 Start address register (FLASH_PCROP2SR)*, *Flash PCROP2 End address register (FLASH_PCROP2ER)*.

In single bank mode (DBANK=0):

- The PCROPx (x = 1,2) area is defined from the address: base address + [PCROPx_STRT x 16] (included) to the address: base address + [(PCROPx_END+1) x 16] (excluded). The minimum PCROP area size is two 2 x double-words (256 bits)

In dual bank mode (DBANK=1)

- The PCROPx (x = 1,2) area is defined from the address: bank “x” base address + [PCROPx_STRT x 0x8] (included) to the address: bank “x” base address + [(PCROPx_END+1) x 0x8] (excluded). The minimum PCROP area size is two double-words (128 bits).

For example, to protect by PCROP from the address 0x0806 2F80 (included) to the address 0x0807 0004 (included):

- if boot in flash is done in Bank 1, FLASH_PCROP1SR and FLASH_PCROP1ER registers must be programmed with:
  - PCROP1_STRT = 0xC5F0.
  - PCROP1_END = 0xE000.
- If the two banks are swapped, the protection must apply to bank 2, and FLASH_PCROP2SR and FLASH_PCROP2ER register must be programmed with:
  - PCROP2_STRT = 0xC5F0.
  - PCROP2_END = 0xE000.

Any read access performed through the D-bus to 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.

For previous example, due to erase by page, all pages from page 0x62 to 0x70 are protected in case of page erase. (All addresses from 0x0806 2000 to 0x080 70FFF can't 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 area, the options programming is launched but PCROP area stays unchanged. On the contrary, it is possible to increase the PCROP area.

When option bit PCROP_RDP is cleared, when the RDP is changing from level 1 to level 0, Full Mass Erase is replaced by Partial Mass Erase in order to keep the PCROP area (refer to *Changing the Read protection level*). In this case, PCROP1/2_STRT and PCROP1/2_END are also not erased.

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

Table 14. PCROP protection[1]

PCROPx registers values (x = 1,2)	PCROP protection area
PCROPx_offset_strt > PCROPx_offset_end	No PCROP area.
PCROPx_offset_strt < PCROPx_offset_end	The area between PCROPx_offset_strt and PCROPx_offset_end is protected. it is possible to write: – PCROPx_offset_strt with a lower value – PCROPx_offset_end with a higher value.

1. When DBANK=1, the minimum PCROP area size is 2xdouble words: PCROPx_offset_strt and PCROPx_offset_end.

When DBANK=0, the minimum PCROP area size is 2x(2xdouble words): PCROPx_offset_strt and PCROPx_offset_end.

When DBANK=1, it is the user's responsibility to make sure no overlapping occurs on the PCROP zones.

3.5.3 Write protection (WRP)

The user area in Flash memory can be protected against unwanted write operations. Depending on the DBANK option bit configuration, it allows either to specify:

- In single bank mode (DBANK=0): four write-protected (WRP) areas can be defined in each bank, with page size (4 KByte) granularity.
- In dual bank mode (DBANK=1): two write-protected (WRP) areas can be defined in each bank, with page (2 KByte) granularity.

Each area is defined by a start page offset and an end page offset related to the physical Flash bank base address. These offsets are defined in the WRP address registers: *Flash WRP area A address register (FLASH_WRP1AR)*, *Flash WRP area B address register (FLASH_WRP1BR)*, *Flash Bank 2 WRP area A address register (FLASH_WRP2AR)*, *Flash Bank 2 WRP area B address register (FLASH_WRP2BR)*.

Dual bank mode (DBANK=1)

The bank "x" WRP "y" area (x=1,2 and y=A,B) is defined from the address: *Bank "x" Base address + [WRPxy_STRT x 0x800] (included)* to the address: *Bank "x" Base address + [(WRPxy_END+1) x 0x800] (excluded)*.

Single Bank mode (DBANK=0)

The WRPx "y" area (x=1,2 and y=A,B) is defined from the address: Base address + [WRPy_STRT x 0x1000] (included) to the address: Base address + [(WRPy_END+1) x 0x1000] (excluded).

For example, to protect by WRP from the address 0x0806 2800 (included) to the address 0x0807 07FF (included):

- if boot in flash is done in Bank 1, FLASH_WRP1AR register must be programmed with:
  - WRP1A_STRT = 0x62.
  - WRP1A_END = 0x70.

  WRP1B_STRT and WRP1B_END in FLASH_WRP1BR can be used instead (area "B" in Bank 1).
- If the two banks are swapped, the protection must apply to bank 2, and FLASH_WRP2AR register must be programmed with:
  - WRP2A_STRT = 0x62.
  - WRP2A_END = 0x70.
  - WRP2A_STRT = 0xC5.
  - WRP2A_END = 0xE0.

  WRP2B_STRT and WRP2B_END in FLASH_WRP2BR can be used instead (area "B in Bank 2).

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 the FLASH_SR register. This flag is also set for any write access to:

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

Note: When the memory read protection level is selected (RDP level = 1), it is not possible to program or erase Flash 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.

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

Note: When DBANK=0, it is the user's responsibility to make sure that no overlapping occurs on the WRP zone.

Table 15. WRP protection

WRP registers values (x=1/2 y= A/B)	WRP protection area
WRPxy_STRT = WRPxy_END	Page WRPxy is protected.
WRPxy_STRT > WRPxy_END	No WRP area.
WRPxy_STRT < WRPxy_END	The pages from WRPxy_STRT to WRPxy_END are protected.

3.5.4 Securable memory area

The Securable memory area defines an area of code which can be executed only once at boot, and never again unless a new reset occurs.

The main purpose of the Securable memory area is to protect a specific part of Flash memory against undesired access. This is a mean to isolate first stage firmware boot together with its sensitive assets (keys) from the rest of the application code. This allows implementing root of trust solution such as secure boot. Securable memory area is located in the Main Flash memory. It is dedicated to executing trusted code. When not secured, the Securable memory behaves like the remainder of Main Flash memory. When secured (the SEC_PROT1 (or SEC_PROT2) bit of the FLASH_CR register set), any attempt to program or erase in a secure memory area generates a write protection error (WRPERR flag is set) and any attempt to read from it generates a read error (RDERR flag is set).

The size of the Securable memory area is defined by the SEC_SIZE1[7:0] (or SEC_SIZE2[7:0]) bitfield of the FLASH_SEC1R (or FLASH_SEC2R) register. It can be modified only in RDP Level 0. Its content is erased upon changing from RDP Level 1 to Level 0, even if it overlaps with PCROP pages.

The securable memory area is defined:

In case of dual bank configuration (DBANK=1):

from the address: Bank1 base address (included) to the address: Bank1 base address + (0x800 * SEC_SIZE1) (excluded) and from the address: Bank2 base address (included) to the address: Bank2 base address + (0x800 * SEC_SIZE2) (excluded) .

In case of single bank configuration DBANK=0:

from the address: Bank base address (included) to the address: Bank base address + (0x1000 * SEC_SIZE1) (excluded).

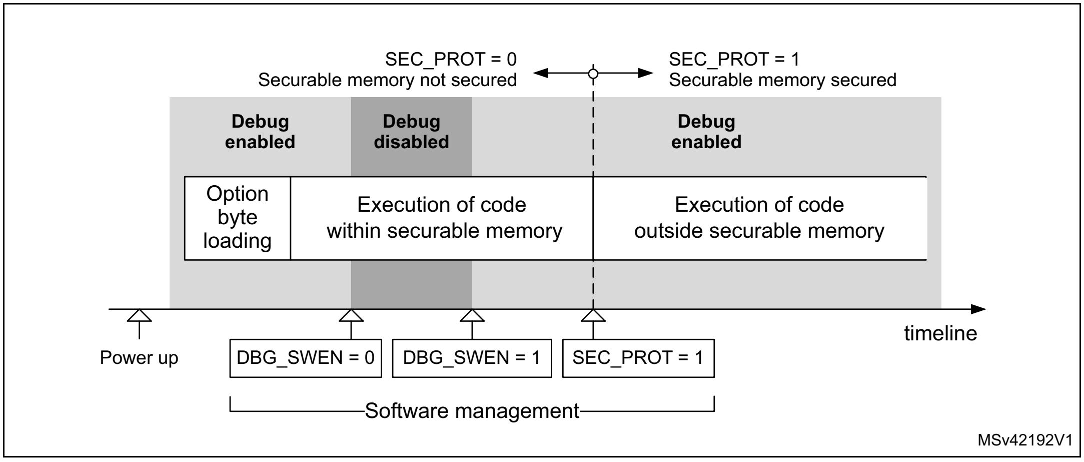
3.5.5 Disabling core debug access

For executing sensitive code or manipulating sensitive data in Securable memory area, the debug access to the core can temporarily be disabled.

In RDP level 2, the debugger is disabled by hardware, but in other RDP levels, the debugger can be disabled by software using the bit DBG_SWEN in the FLASH_ACR register.

Figure 11 gives an example of managing DBG_SWEN and SEC_PROT bits.

Figure 5. Example of disabling core debug access



3.5.6 Forcing boot from Flash memory

To increase the security and establish a chain of trust, the BOOT_LOCK option bit of the FLASH_SEC1R/FLASH_SEC2R register allows forcing the system to boot from the Main Flash memory regardless the other boot options. It is always possible to set the BOOT_LOCK bit. However, it is possible to reset it only when:

- RDP is set to Level 0, or
- RDP is set to Level 1, while Level 0 is requested and a full mass-erase is performed.

3.6 FLASH interrupts

Table 16. Flash interrupt request

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
Read error	RDERR	Write RDERR=1	RDERRIE
ECC correction	ECCC	Write ECCC=1	ECCCIE

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

3.7 FLASH registers

3.7.1 Flash access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0004 0601

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.	DBG_ SWEN	Res.	Res.
													rw		

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_SWEN**: Debug software enable

SW may use this bit to enable/disable the debugger.

0: Debugger disabled

1: Debugger enabled

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **SLEEP_PD**: Flash Power-down mode during Sleep or Low-power sleep mode

This bit determines whether the flash memory is in Power-down mode or Idle mode when the device is in Sleep or Low-power sleep mode.

0: Flash in Idle mode during Sleep and Low-power sleep modes

1: Flash in Power-down mode during Sleep and Low-power sleep modes

Caution: The flash must not be put in power-down while a program or an erase operation is on-going.

Bit 13 **RUN_PD**: Flash Power-down mode during Run or Low-power run mode

This bit is write-protected with FLASH_PDKEYR.

This bit determines whether the flash memory is in Power-down mode or Idle mode when the device is in Run or Low-power run mode. The flash memory can be put in power-down mode only when the code is executed from RAM. The Flash must not be accessed when RUN_PD is set.

0: Flash in Idle mode

1: Flash in Power-down mode

Caution: The flash must not be put in power-down while a program or an erase operation is on-going.

Bit 12 **DCRST**: Data cache reset

0: Data cache is not reset

1: Data cache is reset

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

Bit 11 **ICRST**: Instruction cache reset

0: Instruction cache is not reset

1: Instruction cache is reset

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

Bit 10 **DCEN**: Data cache enable

0: Data cache is disabled

1: Data cache is enabled

Bit 9 **ICEN**: Instruction cache enable

0: Instruction cache is disabled

1: Instruction cache is enabled

Bit 8 **PRFTEN:** Prefetch enable

0: Prefetch disabled

1: Prefetch enabled

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LATENCY[3:0]**: Latency

These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time.

0000: Zero wait state
0001: One wait state
0010: Two wait states
0011: Three wait states
0100: Four wait states
...1111: Fifteen wait states

3.7.2 Flash Power-down key register (FLASH_PDKEYR)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait state, word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDKEYR[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
PDKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDKEYR**: Power-down in Run mode Flash key

The following values must be written consecutively to unlock the RUN_PD bit in FLASH_ACR:

PDKEY1: 0x0415 2637
PDKEY2: 0xFAFB FCFD

3.7.3 Flash key register (FLASH_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait state, word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR[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
KEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEYR**: Flash key

The following values must be written consecutively to unlock the FLACH_CR register allowing flash programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

3.7.4 Flash option key register (FLASH_OPTKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait state, word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEYR[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
OPTKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OPTKEYR**: Option byte key

The following values must be written consecutively to unlock the FLACH_OPTR register allowing option byte programming/erasing operations:
KEY1: 0x0819 2A3B
KEY2: 0x4C5D 6E7F

3.7.5 Flash status register (FLASH_SR)

Address offset: 0x10

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.	BSY
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTV ERR	RD ERR	Res.	Res.	Res.	Res.	FAST ERR	MISS ERR	PGS ERR	SIZ ERR	PGA ERR	WRP ERR	PROG ERR	Res.	OP ERR	EOP
rc_w1	rc_w1					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

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

Bit 16 **BSY**: Busy

This indicates that a Flash operation is in progress. This is set on the beginning of a Flash operation and reset when the operation finishes or when an error occurs.

Bit 15 **OPTVERR**: Option validity error

Set by hardware when the options read may not be the one configured by the user. If option haven’t been properly loaded, OPTVERR is set again after each system reset.

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 (PCROP protection). An interrupt is generated if RDERRIE is set in FLASH_CR.

Cleared by writing 1.

Bits 13:10 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.

Cleared by writing 1.

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

In Fast programming mode, 32 double words must be sent to flash successively, and the new data must be sent to the flash logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.

Cleared by writing 1.

Bit 7 **PGSERR**: Programming sequence error

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. Set also by hardware when PROGERR, SIZERR, PGAERR, WRPERR, MISSERR or FASTERR is set due to a previous programming error.

Set also when trying to perform bank erase when DBANK=0.

Cleared by writing 1.

Bit 6 **SIZERR**: Size error

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).

Cleared by writing 1.

Bit 5 **PGAERR**: Programming alignment error

Set by hardware when the data to program cannot be contained in the same 64-bit Flash memory row in case of standard programming, or if there is a change of page during fast programming.

Cleared by writing 1.

Bit 4 **WRPERR**: Write protection error

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.

Cleared by writing 1.

Bit 3 **PROGERR**: Programming error

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

Cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Operation error

Set by hardware when a Flash memory operation (program / erase) completes unsuccessfully.

This bit is set only if error interrupts are enabled (ERRIE = 1).

Cleared by writing ‘1’.

Bit 0 **EOP**: End of operation

Set by hardware when one or more Flash memory operation (programming / erase) has been completed successfully.

This bit is set only if the end of operation interrupts are enabled (EOPIE = 1).

Cleared by writing 1.

3.7.6 Flash control register (FLASH_CR)

Address offset: 0x14

Reset value: 0xC000 0000

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	SEC_ PROT2	SEC_ PROT1	OBL_ LAUNCH	RD ERRIE	ERR IE	EOP IE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPT STRT	STRT
rs	rs	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**
MER2	Res.	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
rw				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** FLASH_CR Lock

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

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

Bit 29 **SEC_PROT2**: Securable memory area protection bit for bank 2.

This bit is set to lock the access to the Securable area in bank 2. It is set by software when exiting the Securable area, and can only be written once. In case DBANK=0, this bit is useless.

Bit 28 **SEC_PROT1**: Securable memory area protection bit for bank 1.

This bit is set to lock the access to the Securable memory area for bank 1 (or when DBANK=0). It is set by software when exiting the Securable area, and can only be written once.

Bit 27 **OBL_LAUNCH**: Force 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 complete

1: Option byte loading requested

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

This bit enables the interrupt generation when the RDERR bit in the 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 the 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 the 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

This bit triggers an options operation when set.

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

Bit 16 **START**: Start

This bit triggers an erase operation when set. If MER1, MER2 and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition should be forbidden.

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

Bit 15 **MER2**: Bank 2 Mass erase

This bit triggers the bank 2 mass erase (all bank 2 user pages) when set.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **BKER**: Bank erase

DBANK=1

0: Bank 1 is selected for page erase

1: Bank 2 is selected for page erase

DBANK=0

Reserved, must be kept cleared

Bit 10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: Page number selection

These bits select the page to erase:

00000000: page 0
00000001: page 1
...
11111111: page 255

Bit 2 **MER1**: Bank 1 Mass erase

This bit triggers the bank 1 mass erase (all bank 1 user pages) when set.

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

3.7.7 Flash ECC register (FLASH_ECCR)

Address offset: 0x18

Reset value: 0x0000 0000

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	ECCC	ECCD2	ECCC2	Res.	Res.	Res.	ECCC IE	Res.	SYSF_ ECC	BK _ECC	Res.	Res.	ADDR_ECC[18:16]		
rc_w1	rc_w1	rc_w1	rc_w1				rw		r	r			r	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

DBANK=1

Set by hardware when two ECC errors have been detected (only if ECCC/ECCC2/ECCD/ ECCD2 are previously cleared). When this bit is set, a NMI is generated.

Cleared by writing 1.

DBANK=0

Set by hardware when two ECC errors have been detected on 64-bits LSB (bits 63:0) (only if ECCC/ECCC2/ECCD/ ECCD2 are previously cleared). When this bit is set, a NMI is generated.

Cleared by writing 1.

Bit 30 **ECCC**: ECC correction

DBANK=1

Set by hardware when one ECC error has been detected and corrected (only if ECCC/ECCC2/ECCD/ECCD2 are previously cleared). An interrupt is generated if ECCCIE is set.

Cleared by writing 1.

DBANK=0

Set by hardware when one ECC error as been detected and corrected on 64-bits LSB (bits 63:0) (only if ECCC/ECCC2/ECCD/ ECCD2 are previously cleared).

Cleared by writing 1.

Bit 29 **ECCD2**: ECC2 detection

DBANK=0

Set by hardware when two ECC errors have been detected on 64-bits MSB (bits127:64). This bit is set (only if ECCC/ECCC2/ECCD/ECCD2 are previously cleared). When this bit is set, a NMI is generated.

Cleared by writing 1.

DBANK=1

Reserved, must be kept at reset value.

Bit 28 **ECCC2**: ECC correction

DBANK=0

Set by hardware when one ECC error has been detected and corrected on 64-bits MSB (bits127:64). This bit is set (only if ECCC/ECCC2/ECCD/ECCD2 are previously cleared). An interrupt is generated if ECCCIE is set.

Cleared by writing 1.

DBANK=1

Reserved, must be kept at reset value.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **ECCCIE**: ECC correction interrupt enable

0: ECCC interrupt disabled

1: ECCC interrupt enabled.

DBANK=0

This bit enables the interrupt generation when the ECCC or ECCC2 bits in the FLASH_ECCR register are set.

DBANK=1

This bit enables the interrupt generation when the ECCC bit in the FLASH_ECCR register is set.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SYSF_ECC**: System Flash ECC fail

This bit indicates that the ECC error correction or double ECC error detection is located in the System Flash.

Bit 21 **BK_ECC**: ECC fail bank

DBANK=1

This bit indicates which bank is concerned by the ECC error correction or by the double ECC error detection.

0: bank 1

1: bank 2

DBANK=0

If SYSF_ECC is 1, it indicates which bank is concerned by the ECC error

If SYSF_ECC is 0, reserved, must be kept cleared.

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:0 **ADDR_ECC**: ECC fail address

DBANK=0

This bit indicates which address in the Flash memory is concerned by the ECC error correction or by the double ECC error detection.

DBANK=1

This bit indicates which address in the bank is concerned by the ECC error correction or by the double ECC error detection.

3.7.8 Flash option register (FLASH_OPTR)

Address offset: 0x20

Reset value: 0xFXXX XXXX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IRHEN	NRST_MODE [1:0]		n BOOT0	nSW BOOT0	CCM SRAM_ RST	SRAM _PE	nBOOT 1	DBANK	Res.	BFB2	WWDG _SW	IWGD_ STDBY	IWDG_ StOP	IWDG_ SW
	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.	nRST_ SHDW	nRST_ STDBY	nRST_ STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
	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 **IRHEN**: Internal reset holder enable bit

0: Internal resets are propagated as simple pulse on NRST pin

1: Internal resets drives NRST pin low until it is seen as low level

Bits 29:28 **NRST_MODE[1:0]**

00: Reserved

01: Reset Input only: a low level on the NRST pin generates system reset, internal RESET not propagated to the NSRT pin

10: GPIO: standard GPIO pad functionality, only internal RESET possible

11: Bidirectional reset: NRST pin configured in reset input/output mode (legacy mode)

Bit 27 **nBOOT0:** nBOOT0 option bit

0: nBOOT0 = 0

1: nBOOT0 = 1

Bit 26 **nSWBOOT0:** Software BOOT0

0: BOOT0 taken from the option bit nBOOT0

1: BOOT0 taken from PB8/BOOT0 pin

Bit 25 **CCMSRAM_RST:** CCM SRAM Erase when system reset

0: CCM SRAM erased when a system reset occurs

1: CCM SRAM is not erased when a system reset occurs

Bit 24 **SRAM_PE:** SRAM1 and CCM SRAM parity check enable

0: SRAM1 and CCM SRAM parity check enable

1: SRAM1 and CCM SRAM parity check disable

Bit 23 **nBOOT1:** Boot configuration

Together with the BOOT0 pin, this bit selects boot mode from the Flash main memory, SRAM1 or the System memory. Refer to *Section 2.6: Boot configuration*.

Bit 22 **DBANK:**

0: Single bank mode with 128 bits data read width

1: Dual bank mode with 64 bits data

This bit can only be written when PCROPA/B is disabled.

Bit 21 Reserved, must be kept cleared

Bit 20 **BFB2:** Dual-bank boot

0: Dual-bank boot disable

1: Dual-bank boot enable

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 is frozen in Standby mode

1: Independent watchdog counter is running in Standby mode

Bit 17 **IWDG_STOP:** Independent watchdog counter freeze in Stop mode

0: Independent watchdog counter is frozen in Stop mode

1: Independent watchdog counter is running in Stop mode

Bit 16 **IDWG_SW:** Independent watchdog selection

0: Hardware independent watchdog
1: Software independent watchdog

Bit 15 Reserved, must be kept cleared

Bit 14 **nRST_SHDW**

0: Reset generated when entering the Shutdown mode
1: No reset generated when entering the Shutdown mode

Bit 13 **nRST_STDBY**

0: Reset generated when entering the Standby mode
1: No reset generate when entering the Standby mode

Bit 12 **nRST_STOP**

0: Reset generated when entering the Stop mode
1: No reset generated when entering the Stop mode

Bit 11 Reserved, must be kept cleared

Bits10:8 **BOR_LEV:** BOR reset Level

These bits contain the VDD 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

Bits 7:0 **RDP:** Read protection level

0xAA: Level 0, read protection not active
0xCC: Level 2, chip read protection active
Others: Level 1, memories read protection active

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

3.7.9 Flash PCROP1 Start address register (FLASH_PCROP1SR)

Address offset: 0x24

Reset value: 0xFFFF XXXX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, word 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.	PCROP1_STRT[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept cleared

Bits 14:0 **PCROP1_STRT:** PCROP area start offset

DBANK=1

PCROP1_STRT contains the first double-word of the PCROP area for bank1.

DBANK=0

PCROP1_STRT contains the first 2xdouble-word of the PCROP area for all memory.

3.7.10 Flash PCROP1 End address register (FLASH_PCROP1ER)

Address offset: 0x28

Reset value: 0xX000 XXXX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, word, half-word access. PCROP_RDP bit can be accessed with byte access.

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.	PCROP1_END[14:0]														
	rw	rw	rw	rw	rw	rw	rw	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 is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:15 Reserved, must be kept cleared

Bits 14:0 **PCROP1_END:** Bank 1 PCROP area end offset

DBANK=1

PCROP1_END contains the last double-word of the bank 1 PCROP area.

DBANK=0

PCROP1_END contains the last 2x double-word of the first PCROP area in all memory.

3.7.11 Flash Bank 1 WRP area A address register (FLASH_WRP1AR)

Address offset: 0x2C

Reset value: 0x00XX 00XX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, 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.	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 cleared

Bits 22:16 **WRP1A_END:** WRP first area “A” end offset

DBANK=1

WRP1A_END contains the last page of WRP first area in bank1.

DBANK=0

WRP1A_END contains the last page of WRP first area for all memory.

Bits 15:7 Reserved, must be kept cleared

Bits 6:0 **WRP1A_STRT:** WRP first area “A” start offset

DBANK=1

WRP1A_STRT contains the first page of WRP first area for bank1.

DBANK=0

WRP1A_STRT contains the first page of WRP first area for all memory.

3.7.12 Flash Bank 1 WRP area B address register (FLASH_WRP1BR)

Address offset: 0x30

Reset value: 0x00XX 00XX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, 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.	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 cleared

Bits 22:16 **WRP1B_END:** WRP second area “B” end offset
DBANK=1
WRP1B_END contains the last page of the WRP second area for bank1.
DBANK=0
WRP1B_END contains the last page of the WPR second area for all memory.

Bits 15:7 Reserved, must be kept cleared

Bits 6:0 **WRP1B_STRT:** WRP second area “B” start offset
DBANK=1
WRP1B_STRT contains the last page of the WRP second area for bank1.
DBANK=0
WRP1B_STRT contains the last page of the WPR second area for all memory.

3.7.13 Flash PCROP2 Start address register (FLASH_PCROP2SR)

Address offset: 0x44

Reset value: 0xFFFF XXXX

Access: no wait state when no Flash memory operation is on going, word 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.	PCROP2_STRT[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept cleared

Bits 14:0 **PCROP2_STRT:** PCROP area start offset
DBANK=1
PCROP2_STRT contains the first double-word of the PCROP area for bank 2.
DBANK=0
PCROP2_STRT contains the first double-word PCROP area for all memory.

3.7.14 Flash PCROP2 End address register (FLASH_PCROP2ER)

Address offset: 0x48

Reset value: 0x0000 XXXX

Access: no wait state when no Flash memory operation is on going, word 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.	PCROP2_END[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept cleared

Bits 14:0 **PCROP2_END:** PCROP area end offset

DBANK=1

PCROP2_END contains the last double-word of the PCROP area for bank2.

DBANK=0

PCROP2_END contains the last 2xdouble-word of the PCROP area for all the memory.

3.7.15 Flash Bank 2 WRP area A address register (FLASH_WRP2AR)

Address offset: 0x4C

Reset value: 0x00XX 00XX

Access: no wait state when no Flash memory operation is on going, 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.	WRP2A_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.	WRP2A_STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept cleared

Bits 22:16 **WRP2A_END:** WRP first area “A” end offset

DBANK=1

WRP2A_END contains the last page of the WRP first area for bank2.

DBANK=0

WRP2A_END contains the last page of the WRP third area for all memory.

Bits 15:7 Reserved, must be kept cleared

Bits 6:0 **WRP2A_STRT:** WRP first area “A” start offset

DBANK=1

WRP2A_STRT contains the first page of the WRP first area for bank2.

DBANK=0

WRP2A_STRT contains the first page of the WRP third area for all memory.

3.7.16 Flash Bank 2 WRP area B address register (FLASH_WRP2BR)

Address offset: 0x50

Reset value: 0x00XX 00XX

Access: no wait state when no Flash memory operation is on going, 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.	WRP2B_END[7:0]						
									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.	WRP2B_STRT[7:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept cleared

Bits 22:16 **WRP2B_END:** WRP second area “B” end offset
DBANK=1
WRP2B_END contains the last page of the WRP second area for bank2.
DBANK=0
WRP2B_END contains the last page of the WRP fourth area for all memory.

Bits 15:7 Reserved, must be kept cleared

Bits 6:0 **WRP2B_STRT:** WRP second area “B” start offset
DBANK=1
WRP2B_STRT contains the first page of the WRP second area for bank2.
DBANK=0
WRP2B_STRT contains the first page of the WRP second area for all memory.

3.7.17 Flash Securable area bank1 register (FLASH_SEC1R)

Address offset: 0x70

Reset value: 0xFFFX FFXX

Access: no wait state when no Flash memory operation is on going, 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.	BOOT_LOCK
															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.	SEC_SIZE1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept cleared

Bit 16 **BOOT_LOCK:** used to force boot from user Flash area.
0: Boot based on the pad/option bit configuration
1: Boot forced from Main Flash memory

Bits 15:8 Reserved, must be kept cleared

Bit 7:0 **SEC_SIZE1[7:0]**: sets the number of pages used in the bank 1 Securable memory area.
Securable area starts at @ 0x0800 0000 and its size is SEC_SIZE1 * page size.
This field can be changed in level0 only.
Any attempt to modify in level1 silently fails, and does not change register value.

3.7.18 Flash Securable area bank2 register (FLASH_SEC2R)

Address offset: 0x74

Reset value: 0xFFFF FFXX

Access: no wait state when no Flash memory operation is on going, 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.	SEC_SIZE2							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept cleared

Bit 7:0 **SEC_SIZE2**: sets the number of pages used in the bank 2 Securable memory area.
Securable area starts at @ 0x0804 0000 and its size is SEC_SIZE2 * page size.
When DBANK=0, this field is not usefull.
This field can be changed in level0 only. Any attempt to modify in level1 silently fails, and does change register value.

3.7.19 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
0x00	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_SWEN	Res.	Res.	Res.	SLEEP_PD	RUN_PD	DCRST	ICRST	DCEN	ICEN	PRFTEN	Res.	Res.	Res.	Res.	LATENCY [3:0]			
	Reset value														1				0	0	0	0	1	1	0					0	0	0	1
0x04	FLASH_ PDKEYR	PDKEYR[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	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
0x0C	FLASH_OPT KEYR	OPTKEYR[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
0x10	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSY	OPTVERR	RDERR	Res.	Res.	Res.	Res.	FASTERR	MISERR	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value																0	0	0					0	0	0	0	0	0	0		0	0
0x14	FLASH_CR	LOCK	OPTLOCK	SEC_PROT2	SEC_PROT1	OBL_LAUNCH	RDERRIE	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPTSTRT	STRT	MER2	Res.	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
	Reset value	1	1	1	1	0	0	0	0						0	0	0	0				0		0	0	0	0	0	0	0	0	0	0
0x18	FLASH_ECCR	ECCD	ECCC	ECCD2	ECCC2	Res.	Res.	Res.	ECCCIE	Res.	SYSF_ECC	BK_ECC	Res.	Res.	ADDR_ECC[18:0]																		
	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
0x20	FLASH_OPTR	Res.	Res.	Res.	Res.	nBOOT0	nSWBOOT0	CCMSRAM_RST	SRAM_PE	nBOOT1	DBANK	Res.	BFB2	WWDG_SW	IWDG_STBY	IWDG_STOP	IWDG_SW	Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	Res.	BOR_ LEV[2:0]			RDP[7:0]							
	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
0x24	FLASH_ PCROP1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_STRT[14:0]														
	Reset value																		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x28	FLASH_ PCROP1ER	PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_END[14:0]														
	Reset value	x																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x2C	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										X	X	X	X	X	X	X										X	X	X	X	X	X	X

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
0x30	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										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x44	FLASH_ PCROP2SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2_STRT[14:0]														
	Reset value																		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x48	FLASH_ PCROP2ER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2_END[14:0]														
	Reset value																		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x4C	FLASH_ WRP2AR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_STRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x50	FLASH_ WRP2BR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_END[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_STRT[7:0]							
	Reset value									X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X
0x70	FLASH_ SEC1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_SIZE1[7:0]							
	Reset value																X									X	X	X	X	X	X	X	X
0x74	FLASH_ SEC2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_SIZE2[7:0]							
	Reset value																									X	X	X	X	X	X	X	X

Refer to *Section 2.2 on page 81* for the register boundary addresses.

4 Embedded Flash memory (FLASH) for category 4 devices

4.1 Introduction

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

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

4.2 FLASH main features

- Up to 512 Kbyte of Flash memory (single bank).
- Flash memory read operations with 64 bits data width
- Page erase and mass erase

Flash memory interface features:

- Flash memory read operations
- Flash memory program/erase operations
- Read protection activated by option (RDP)
- 2 Write protection areas selected by option
- Proprietary code read protection areas defined by option
- Securable memory areas defined by option
- Prefetch on ICODE
- Instruction Cache: 32 cache lines of 4 x 64 bits on ICode (1 KB RAM)
- Data Cache: 8 cache lines of 4 x 64 bits on DCode (256B RAM)
- Error Code Correction ECC: 8 bits per 64-bit double-word
  - 8 + 64 = 72 bits, 2 bits detection, 1 bit correction
- Option byte loader
- Low-power mode

4.3 FLASH functional description

4.3.1 Flash memory organization

The Flash memory has the following main features:

- Capacity up to 512 Kbytes (read width of 64-bits)
- 512 KB organized in one single bank for main memory
- Page size of 2 Kbyte
- 72 bits wide data read (64 bits plus 8 ECC bits)
- Page and Mass erase
- Each page is composed of 8 rows of 256 bytes
- An Information block containing:
  - System memory from which the device boots in System memory boot mode. The area is reserved for use by STMicroelectronics and contains the boot loader that is used to reprogram the Flash memory through one of the following interfaces: USART, SPI, I2C, USB. It is programmed by STMicroelectronics when the device is manufactured, and protected against spurious write/erase operations. For further details, please refer to the AN2606 available from *www.st.com*.
  - 1 Kbyte (128 double word) OTP (one-time programmable) bytes 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 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 *Table 28*.

Table 18. Flash module organization (64 bits read width)

Flash area	Flash memory addresses	Size (bytes)	Name
Main memory	0x0800 0000 - 0x0800 07FF	2 K	Page 0
	0x0800 0800 - 0x0800 0FFF	2 K	Page 1
	0x0800 1000 - 0x0800 17FF	2 K	Page 2
	0x0800 1800 - 0x0800 1FFF	2 K	Page 3
	- - - -	- - - -	- - - -
	0x0807 F800 - 0x0807 FFFF	2 K	Page 255
Information block	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	0x1FFF 7800 - 0x1FFF 782F	48	Option bytes

4.3.2 Error code correction (ECC)

Data in Flash memory are 72-bits words: 8 bits are added per double word (64 bits). The ECC mechanism supports:

- One error detection and correction
- Two errors detection

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

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

When an ECC error is detected, the address of the failing double word is saved in ADDR_ECC[20:0] in the FLASH_ECCR register. ADDR_ECC[2:0] are always cleared.

When ECCC or ECCD is set, ADDR_ECC 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 two errors detection 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.

4.3.3 Read access latency

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the *Flash access control register (FLASH_ACR)* according to the frequency of the CPU clock (HCLK) and the internal voltage range of the device V_{CORE}. Refer to *Section 6.1.5: Dynamic voltage scaling management*. *Table 19* shows the correspondence between wait states and CPU clock frequency.

Table 19. Number of wait states according to CPU clock (HCLK) frequency

Wait states (WS) (LATENCY)	HCLK (MHz)		
	V_{CORE} Range 1 boost mode	V_{CORE} Range 1 normal mode	V_{CORE} Range 2
0 WS (1 CPU cycles)	≤ 34	≤ 30	≤ 12
1 WS (2 CPU cycles)	≤ 68	≤ 60	≤ 24
2 WS (3 CPU cycles)	≤ 102	≤ 90	≤ 26
3 WS (4 CPU cycles)	≤ 136	≤ 120	-
4 WS (5 CPU cycles)	≤ 170	≤ 150	-

After reset, the CPU clock frequency is 16 MHz and 1 wait state (WS) is configured in the FLASH_ACR register.

When changing the CPU frequency, the following software sequences must be applied in order to tune the number of wait states needed to access the Flash memory:

Increasing the CPU frequency:

1. Program the new number of wait states to the LATENCY bits in the *Flash access control register (FLASH_ACR)*.
2. Check that the new number of wait states is taken into account to access the Flash memory by reading the FLASH_ACR register.
3. Analyze the change of CPU frequency change caused either by:
   - changing clock source defined by SW bits in RCC_CFGR register
   - or by CPU clock prescaller defined by HPRE bits in RCC_CFGR

   If some of above two steps decreases the CPU frequency, firstly perform this step and then the rest. Otherwise modify The CPU clock source by writing the SW bits in the RCC_CFGR register and then (if needed) modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.
4. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register.

Decreasing the CPU frequency:

1. Modify the CPU clock source by writing the SW bits in the RCC_CFGR register.
2. If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.
3. Analyze the change of CPU frequency change caused either by:
   - changing clock source defined by SW bits in RCC_CFGR register
   - or by CPU clock prescaller defined by HPRE bits in RCC_CFGR

   If some of above two steps increases the CPU frequency, firstly perform another step and then this step. Otherwise modify The CPU clock source by writing the SW bits in the RCC_CFGR register and then (if needed) modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.
4. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register.
5. Program the new number of wait states to the LATENCY bits in *Flash access control register (FLASH_ACR)*.
6. Check that the new number of wait states is used to access the Flash memory by reading the FLASH_ACR register.

4.3.4 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 FPU processors. It balances the inherent performance advantage of the Arm® Cortex®-M4 with FPU over Flash memory technologies, which normally requires 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 which 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 Flash memory at a CPU frequency up to 170 MHz.

Instruction prefetch

The Cortex®-M4 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.

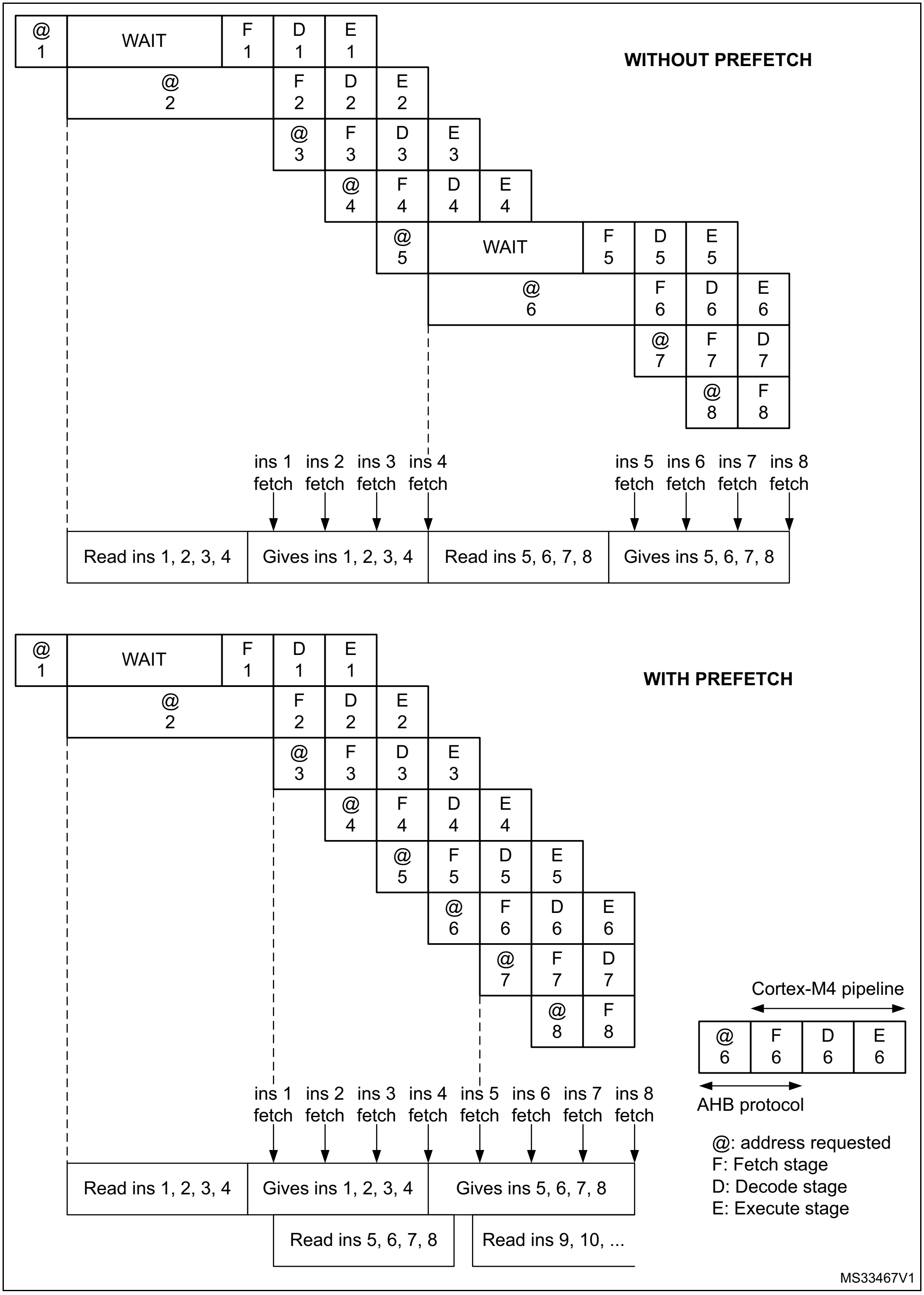
Each Flash memory read operation provides 64 bits from either two instructions of 32 bits or four instructions of 16 bits depending on the launched program. This 64-bits current instruction line is saved in a current buffer, and in case of sequential code, at least two CPU cycles are needed to execute the previous read instruction line.

Prefetch on the 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 the PRFTEN bit in the *Flash access control register (FLASH_ACR)*. This feature is useful if at least one wait state is needed to access the Flash memory.

Figure 9 shows the execution of sequential 16-bit instructions with and without prefetch when 3 WS are needed to access the Flash memory.

Figure 6. Sequential 16-bit instructions execution (64-bit read data width)



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 flash access is performed.

Instruction cache memory (I-Cache)

To limit the time lost due to jumps, it is possible to retain 32 lines of 4 x 64 bits in an instruction cache memory.This feature can be enabled by setting the instruction cache enable (ICEN) bit in the *Flash access control register (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 enable after system reset.

Data cache memory (D-Cache)

Literal pools are fetched from Flash memory through the DCode bus during the execution stage of the CPU pipeline. Each DCode bus read access fetches 64 bits which 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 databus DCode have priority over accesses through the AHB instruction bus ICode.

If some literal pools are frequently used, the data cache memory can be enabled by setting the data cache enable (DCEN) bit in the *Flash access control register (FLASH_ACR)*. This feature works like the instruction cache memory, but the retained data size is limited to 8 rows of 4*64 bits.

The Data cache memory is enable after system reset.

Note: The D-Cache is active only when data is requested by the CPU (not by DMA1 and DMA2). Data in option bytes block are not cacheable.

4.3.5 Flash program and erase operations

The STM32G4 Series 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 boot loader to load the user application 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 (I/Os, USB, CAN, UART, I^2C, SPI, etc.) 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 has to 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.

The Flash erase and programming is only possible in the voltage scaling range 1. The VOS[1:0] bits in the PWR_CR1 must be programmed to 01b.

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 has completed.

Unlocking the Flash memory

After reset, write is not allowed in the *Flash control register (FLASH_CR)* to protect the Flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence is used to unlock this register:

1. Write KEY1 = 0x45670123 in the *Flash key register (FLASH_KEYR)*
2. Write KEY2 = 0xCDEF89AB in the FLASH_KEYR register.

Any wrong sequence locks up the FLASH_CR 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 FLASH_CR register can be locked again by software by setting the LOCK bit in the FLASH_CR register.

Note: The FLASH_CR register cannot be written when the BSY bit in the Flash status register (FLASH_SR) is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

4.3.6 Flash main memory erase sequences

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

Page erase

To erase a page, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH_SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PER bit and select the page to erase (PNB).
4. Set the STRT bit in the FLASH_CR register.
5. Wait for the BSY bit to be cleared in the FLASH_SR register.

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

If the page erase is part of write-protected area (by WRP or PCROP), WRPERR is set and the page erase request is aborted.

Mass erase

To perform a Mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the MER1 bit in the Flash control register (FLASH_CR).
4. Set the STRT bit in the FLACH_CR register.
5. Wait for the BSY bit to be cleared in the Flash status register (FLASH_SR).

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

If the Flasb memory contains a write-protected area (by WRP or PCROP), WRPERR is set and the mass erase request is aborted.

4.3.7 Flash main memory programming sequences

The Flash memory is programmed 72 bits at a time (64 bits + 8 bits ECC).

Programming in a previously programmed address is not allowed except if the data to write is full zero, and any attempt sets PROGERR flag in the *Flash status register (FLASH_SR)*.

It is only possible to program double word (2 x 32-bit data).

- Any attempt to write byte or half-word sets SIZERR flag in the FLASH_SR register.
- Any attempt to write a double word which is not aligned with a double word address sets PGAERR flag in the FLASH_SR register.

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 the BSY bit in the *Flash status register (FLASH_SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PG bit in the *Flash control register (FLASH_CR)*.
4. Perform the data write operation at the desired memory address, inside main memory block or OTP area. Only double word can be programmed.
   – Write a first word in an address aligned with double word
   – Write the second word
5. Wait until the BSY bit is cleared in the FLASH_SR register.
6. Check that EOP flag is set in the FLASH_SR register (meaning that the programming operation has succeed), and clear it by software.
7. Clear the PG bit in the FLASH_SR register if there no more programming request anymore.

Note: When the flash interface has received a good sequence (a double word), programming is automatically launched and BSY bit is set. The internal oscillator HSI16 (16 MHz) is enabled *automatically when PG bit is set, and disabled automatically when PG bit is cleared, except if the HSI16 is previously enabled with HSION in 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.

Fast programming

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

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

The Flash main memory programming sequence in standard mode is as follows:

1. Perform a mass erase. If not, PGSERR is set.
2. Check that no Flash main memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH_SR)*.
3. Check and clear all error programming flag due to a previous programming.
4. Set the FSTPG bit in *Flash control register (FLASH_CR)*.
5. Write the 32 double words to program a row. Only double words can be programmed:
   - Write a first word in an address aligned with double word
   - Write the second word.
6. Wait until the BSY bit is cleared in the FLASH_SR register.
7. Check that EOP flag is set in the FLASH_SR register (meaning that the programming operation has succeed), and clear it by software.
8. Clear the FSTPG bit in the FLASH_SR register if there no more programming request anymore.

Note: If the flash is attempted to be written in Fast programming mode while a read operation is on going, the programming is aborted without any system notification (no error flag is set).

When the Flash interface has received 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 has been programmed or in case of error. The internal oscillator HSI16 (16 MHz) is enabled automatically when FSTPG bit is set, and disabled automatically when FSTPG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

The 32 double word must be written successively. The high voltage is kept on the flash for all the programming. Maximum time between two double words write requests is the time programming (around 2 x 25us). If a second double word arrives after this time programming, fast programming is interrupted and MISSERR is set.

High voltage mustn't exceed 8 ms for a full row between 2 erases. This is guaranteed by the sequence of 32 double words successively written with a clock system greater or equal to 8MHz. An internal time-out counter counts 7ms when Fast programming is set and stops the programming when time-out is over. In this case the FASTERR bit 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

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 is not previously erased (except if the value to program is full zero).
- **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 doesn't belong to the same double word address.
  - In fast programming: the data to program doesn't 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: MER1, and PER are not cleared when PG or FSTPG is set.
  - In the fast programming sequence: the Mass erase is not performed before setting FSTPG bit.
  - In the mass erase sequence: PG, FSTPG, and PER are not cleared when MER1 is set.
  - In the page erase sequence: PG, FSTPG, MER1 are not cleared when PER is set.
  - PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR or PGSERR is set due to a previous programming error.
- **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 or in a Securable memory area.
  - Attempt to perform an erase when one page or more is protected by WRP or PCROP.
  - The debug features are connected or the boot is executed from SRAM or from System flash when the read protection (RDP) is set to Level 1.
  - Attempt to modify the option bytes when the read 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 programmation 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 7ms which generates a time-out detection.
- When the fast programming has been 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 the FLASH_SR register:

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 the *Flash status register (FLASH_SR)*, an interrupt is generated and the operation error flag OPERR is set in the FLASH_SR register.

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 requests.

Programming and caches

If a Flash memory write access concerns some data in the data cache, the Flash write access modifies the data in the Flash memory and the data in the cache.

If an erase operation in Flash memory also concerns data in the data or instruction cache, you have to make sure that these data are rewritten before they are accessed during code execution. If this cannot be done safely, it is recommended to flush the caches by setting the DCRST and ICRST bits in the *Flash access control register (FLASH_ACR)*.

Note: The I/D cache should be flushed only when it is disabled (I/DCEN = 0).

4.4 FLASH option bytes

4.4.1 Option bytes description

The option bytes are configured by the end user depending on the application requirements. As a configuration example, the watchdog may be selected in hardware or software mode (refer to *Section 5.4.2: Option bytes programming*).

A double word is split up as follows in the option bytes:

Table 20. Option byte format

63-24	23-16	15 -8	7-0	31-24	23-16	15 -8	7-0
Complemented option byte 3	Complemented option byte 2	Complemented option byte 1	Complemented option byte 0	Option byte 3	Option byte 2	Option byte 1	Option byte 0

The organization of these bytes inside the information block is as shown in *Table 31: Option byte organization*.

The option bytes can be read from the memory locations listed in *Table 31: Option byte organization* or from the Option byte registers:

- *Flash option register (FLASH_OPTR)*
- *Flash PCROP1 Start address register (FLASH_PCROP1SR)*
- *Flash PCROP1 End address register (FLASH_PCROP1ER)*
- *Flash WRP area A address register (FLASH_WRP1AR)*
- *Flash WRP area B address register (FLASH_WRP1BR)*

Table 21. Option byte organization

Address	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
1FFF7800	$\overline{\text{USER OPT}}$			$\overline{\text{RDP}}$	USER OPT			RDP
1FFF7808	Unused		$\overline{\text{Unused and PCROP1\_STRT[15:0]}}$		Unused		Unused and PCROP1_STRT[15:0]	
1FFF7810	$\overline{\text{PCROP\_RDP and Unused}}$		$\overline{\text{Unused and PCROP1\_END[15:0]}}$		PCROP_RDP and Unused		Unused and PCROP1_END[15:0]	
1FFF7818	$\overline{\text{Unused}}$	$\overline{\text{WRP1A\_END [7:0]}}$	$\overline{\text{Unused}}$	$\overline{\text{WRP1A\_STRT [7:0]}}$	Unused	WRP1A_ END [7:0]	Unused	WRP1A _STRT [7:0]
1FFF7820	$\overline{\text{Unused}}$	$\overline{\text{WRP2A\_END [7:0]}}$	$\overline{\text{Unused}}$	$\overline{\text{WRP2A\_STRT [7:0]}}$	Unused	WRP2A_ END [7:0]	Unused	WRP2A _STRT [7:0]
1FFF7828	$\overline{\text{Unused}}$	$\overline{\text{BOOT\_LOCK}}$	$\overline{\text{Unused}}$	$\overline{\text{SEC\_SIZE1}}$	Unused	BOOT_ LOCK	Unused	SEC_ SIZE1

User and read protection option bytes

Flash memory address: 0x1FFF 7800

ST production value: 0xFFEF F8AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IRH_EN	NRST_MODE		nBOOT0	nSWBOOT0	CCMSRAM_RST	SRAM_PE	nBOOT1	PB4_PUPEN	Res.	Res.	WWDG_SW	IWGD_STDBY	IWDG_STOP	IWDG_SW
	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.	nRST_SHDW	nRST_STDBY	nRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
	r	r	r		r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **IRH_IN**: Internal reset holder for PG10

0: IRH disabled

1: IRH enabled

Bits 29:28 **NRST_MODE**: PG10 pad mode

00: Reset Input/Output

01: Reset Input only

10: GPIO

11: Reset Input/Output

Bit 27 **nBOOT0:** nBOOT0 option bit

0: nBOOT0 = 0

1: nBOOT0 = 1

Bit 26 **nSWBOOT0:** Software BOOT0

0: BOOT0 taken from the option bit nBOOT0

1: BOOT0 taken from PB8/BOOT0 pin

Bit 25 **CCMSRAM_RST:** CCM SRAM erase when system reset

0: CCM SRAM erased when a system reset occurs

1: CCM SRAM is not erased when a system reset occurs

Bit 24 **SRAM_PE:** SRAM1 and CCM SRAM parity check enable

0: SRAM1 and CCM SRAM parity check enable

1: SRAM1 and CCM SRAM parity check disable

Bit 23 **nBOOT1:** Boot configuration

Together with the BOOT0 pin, this bit selects boot mode from the Flash main memory, SRAM1 or the System memory. Refer to *Section 2.6: Boot configuration*.

Bit 22 **PB4_PUPEN:** PB4 pull-up enable

0: USB power delivery dead-battery enabled/ TDI pull-up deactivated

1: USB power delivery dead-battery disabled/ TDI pull-up activated

Note: Only for Category 4 devices (otherwise Reserved)

Bits 21:20 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 is frozen in Standby mode

1: Independent watchdog counter is running in Standby mode

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

Bit 16 **IDWG_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**
0: Reset generated when entering the Shutdown mode
1: No reset generated when entering the Shutdown mode

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

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

Bit 11 Reserved, must be kept at reset value.

Bits10:8 **BOR_LEV:** BOR reset Level
These bits contain the VDD 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

Bits 7:0 **RDP:** Read protection level
0xAA: Level 0, read protection not active
0xCC: Level 2, chip read protection active
Others: Level 1, memories read protection active

PCROP1 Start address option bytes

Flash memory address: 0x1FFF 7808

ST production value: 0xFFFF FFFF

31	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
PCROP1_STRT[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 **PCROP1_STRT:** PCROP area start offset
PCROP1_STRT contains the first double-word of the PCROP area for bank1.

PCROP1 End address option bytes

Flash memory address: 0x1FFF 7810

ST production value: 0x00FF 0000

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.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCROP1_END[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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 is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **PCROP1_END:** Bank 1 PCROP area end offset

PCROP1_END contains the last double-word of the PCROP area.

WRP1 Area A address option bytes

Flash memory address: 0x1FFF 7818

ST production value: 0xFF00 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_END[7:0]							
								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.	WRP1A_STRT[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRP1A_END:** WRP first area “A” end offset

WRP1A_END contains the last page of WRP first area.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **WRP1A_STRT:** WRP first area “A” start offset

WRP1A_STRT contains the first page of WRP first area.

WRP2 Area A address option bytes

Flash memory address: 0x1FFF 7820

ST production value: 0xFF00 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_END[7:0]							
								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.	WRP2A_STRT[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRP2A_END:** WRP second area “B” end offset
WRP2A_END contains the last page of the WRP second area.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **WRP2A_STRT:** WRP second area start offset
WRP2A_STRT contains the last page of the WRP second area.

Securable memory area option bytes

Flash memory address: 0x1FFF7828

ST production value: 0xFF00FE00

31	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.	BOOT_LOCK
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_SIZE1[8:0]								
							r	r	r	r	r	r	r	r	r

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

Bit 16 **BOOT_LOCK** used to force boot from user area
0: Boot based on the pad/option bit configuration
1: Boot forced from Main Flash memory

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **SEC_SIZE1[8:0]:** Securable memory area size
Contains the number of Securable Flash memory pages

4.4.2 Option bytes programming

After reset, the options related bits in the *Flash control register (FLASH_CR)* are write-protected. To run any operation on the option bytes page, the option lock bit OPTLOCK in the *Flash control register (FLASH_CR)* must be cleared. The following sequence is used to unlock this register:

1. Unlock the FLASH_CR with the LOCK clearing sequence (refer to *Unlocking the Flash memory*).
2. Write OPTKEY1 = 0x08192A3B in the *Flash option key register (FLASH_OPTKEYR)*.
3. Write OPTKEY2 = 0x4C5D6E7F in the FLASH_OPTKEYR register.

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 too.

Modifying 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. Check that no Flash memory operation is on going by checking the BSY bit in the *Flash status register (FLASH_SR)*.
2. Clear OPTLOCK option lock bit with the clearing sequence described above.
3. Write the desired options value in the options registers: *Flash option register (FLASH_OPTR)*, *Flash PCROP1 Start address register (FLASH_PCROP1SR)*, *Flash PCROP1 End address register (FLASH_PCROP1ER)*, *Flash WRP area A address register (FLASH_WRP1AR)*, *Flash WRP area B address register (FLASH_WRP1BR)*,
4. Set the Options Start bit OPTSTRT in the *Flash control register (FLASH_CR)*.
5. Wait for the BSY bit to be cleared.

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

Option byte loading

After the BSY bit is cleared, all new options are updated into the flash but they are not applied to the system. They have effect on the system when they are loaded. Option bytes loading (OBL) is performed in two cases:

- when OBL_LAUNCH bit is set in the *Flash control register (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 cannot be read with 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 to check the loading has correctly taken place.

During option byte loading, the options are read by double word with ECC. 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:

- For USR OPT option, the value of mismatch is all options at ‘1’, except for BOR_LEV which 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”

On system reset rising, internal option registers are copied into option registers which can be read and written by software (FLASH_OPTR, FLASH_PCROP1SR, FLASH_PCROP1ER, FLASH_WRP1AR, FLASH_WRP1BR). These registers are also used to modify options. If these registers are not modified by user, they reflects the options states of the system. See *Section : Modifying user options* for more details.

4.5 FLASH memory protection

The Flash main memory can be protected against external accesses with the Read protection (RDP). The pages of the Flash memory can also be protected against unwanted write due to loss of program counter contexts. The write-protection (WRP) granularity is one page (2 Kbyte). Apart of the flash memory can also be protected against read and write from third parties (PCROP). The PCROP granularity is double word (64-bit).

4.5.1 Read protection (RDP)

The read protection is activated by setting the RDP option byte and then, by applying a system reset to reload the new RDP option byte. The read protection protects to the Flash main memory, the option bytes, the backup registers (RTC_BKPxR in the RTC) and the CCM SRAM.

Note: If the read protection is set while the debugger is still connected through JTAG/SWD, apply a POR (power-on reset) instead of a system reset.

There are three levels of read 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 *Table 32*.

Table 22. Flash memory read protection status

RDP byte value	RDP complement value	Read protection level
0xAA	0x55	Level 0 (production value)
Any value except 0xAA or 0xCC	Any value (not necessarily complementary) except 0x55 and 0x33	Level 1
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 Flash main memory area are possible. The option bytes, the CCM SRAM and the backup registers are also accessible by all operations.

Level 1: Read protection

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

- **User mode:** Code executing in user mode (**Boot Flash**) can access Flash main memory, option bytes, CCM SRAM and backup registers with all operations.
- **Debug, boot RAM and boot loader modes:** In debug mode or when code is running from boot RAM or boot loader, the Flash main memory, the backup registers (RTC_BKPxR in the RTC) and the CCM SRAM are totally inaccessible. In these modes, a read or write access to the Flash generates a bus error and a Hard Fault interrupt.

Caution: In case the Level 1 is configured and no PCROP area is 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 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 Cortex®-M4 debug port, the boot from RAM (boot RAM mode) and the boot from System memory (boot loader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the Flash Main memory. On the contrary, only read operations can be performed on the option bytes.

Option bytes cannot be programmed nor erased. Thus, the level 2 cannot be removed at all: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag WRPERR is set in the Flash_SR register and an interrupt can be generated.

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.

Changing the Read 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 either directly from level 0 or from level 1. Once in level 2, it is no more possible to modify the Read protection level.

When the RDP is reprogrammed to the value 0xAA to move from Level 1 to Level 0, a mass erase of the Flash main memory is performed if PCROP_RDP is set in the *Flash PCROP1 End address register (FLASH_PCROP1ER)*. The backup registers (RTC_BKPxR in the RTC) and the CCM SRAM are also erased. The user options except PCROP protection are set to their previous values copied from FLASH_OPTR, FLASH_WRPxyR (x=1 and y =A or B). PCROP is disable. The OTP area is not affected by mass erase and remains unchanged.

If the bit PCROP_RDP is cleared in the FLASH_PCROP1ER, 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_PCROPxSR and FLASH_PCROPxER registers (x=1).

Note: Full Mass Erase or Partial Mass Erase is performed only when Level 1 is active and Level 0 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 control register.

Figure 7. Changing the read protection (RDP) level

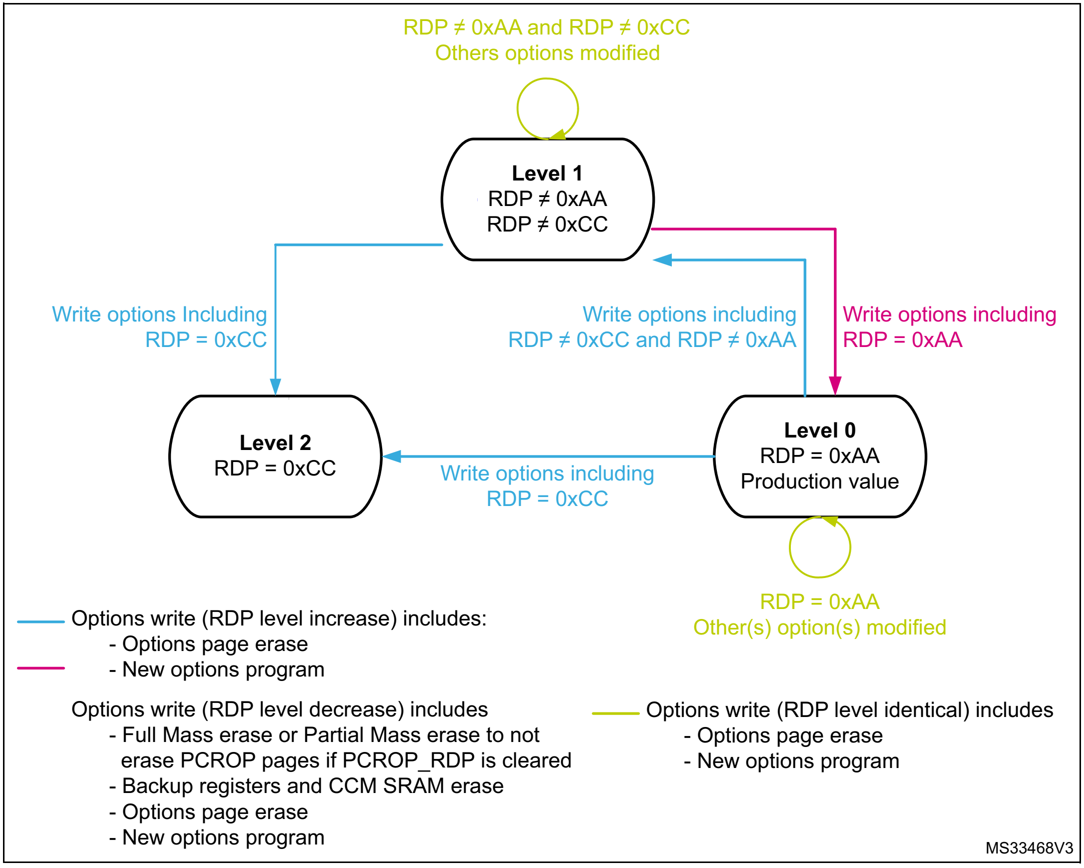


Table 23. Access status versus protection level and execution modes

Area	Protection level	User execution (BootFromFlash)			Debug/ BootFromRam/ BootFromLoader[1]		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No[3]
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory [2]	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A

Table 23. Access status versus protection level and execution modes (continued)

Area	Protection level	User execution (BootFromFlash)			Debug/ BootFromRam/ BootFromLoader[1]		
		Read	Write	Erase	Read	Write	Erase
Option bytes	1	Yes	Yes[3]	Yes	Yes	Yes[3]	Yes
	2	Yes	No	No	N/A	N/A	N/A
OTP	1	Yes	Yes[4]	N/A	No	No	N/A
	2	Yes	Yes[4]	N/A	N/A	N/A	N/A
Backup registers	1	Yes	Yes	N/A	No	No	No[5]
	2	Yes	Yes	N/A	N/A	N/A	N/A
CCM SRAM	1	Yes	Yes	N/A	No	No	No[6]
	2	Yes	Yes	N/A	N/A	N/A	N/A

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 main memory is erased when the RDP option byte is programmed with all level protections disabled (0xAA).
4. OTP can only be written once.
5. The backup registers are erased when RDP changes from level 1 to level 0.
6. The CCM SRAM is erased when RDP changes from level 1 to level 0.

4.5.2 Proprietary code readout protection (PCROP)

Apart 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, as an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. The PCROP area has a double word (64-bit) granularity. An additional option bit (PCROP_RDP) allows to select if the PCROP area is erased or not when the RDP protection is changed from Level 1 to Level 0 (refer to *Changing the Read protection level*).

Each PCROP area is defined by a start page offset and an end page offset related to the physical Flash address. These offsets are defined in the PCROP address registers *Flash PCROP1 Start address register (FLASH_PCROP1SR)*, *Flash PCROP1 End address register (FLASH_PCROP1ER)*.

- The PCROPx (x = 1) area is defined from the address: Flash memory base address + [PCROPx_STRT x 0x8] (included) to the address: Flash memory base address + [(PCROPx_END+1) x 0x8] (excluded). The minimum PCROP area size is two double-words (128 bits).

For example, to protect by PCROP from the address 0x0806 2F80 (included) to the address 0x0807 0004 (included):

- if boot in flash is selected, FLASH_PCROP1SR and FLASH_PCROP1ER registers must be programmed with:
  - PCROP1_STRT = 0xC5F0.
  - PCROP1_END = 0xE000.

Any read access performed through the D-bus to a PCROP protected area triggers 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.

For previous example, due to erase by page, all pages from page 0xC5 to 0xE0 are protected in case of page erase. (All addresses from 0x0806 2800 to 0x0807 07FF can’t 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 area, the options programming is launched but PCROP area stays unchanged. On the contrary, it is possible to increase the PCROP area.

When option bit PCROP_RDP is cleared, when the RDP is changing from level 1 to level 0, Full Mass Erase is replaced by Partial Mass Erase in order to keep the PCROP area (refer to *Changing the Read protection level*). In this case, PCROP1_STRT and PCROP1_END are also not erased.

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

Table 24. PCROP protection[1]

PCROPx registers values (x = 1)	PCROP protection area
PCROPx_offset_strt > PCROPx_offset_end	No PCROP area.
PCROPx_offset_strt < PCROPx_offset_end	The area between PCROPx_offset_strt and PCROPx_offset_end is protected. it is possible to write: – PCROPx_offset_strt with a lower value – PCROPx_offset_end with a higher value.

1. The minimum PCROP area size is 2xdouble words: PCROPx_offset_strt and PCROPx_offset_end.

4.5.3 Write protection (WRP)

The user area in Flash memory can be protected against unwanted write operations. It allows either to specify:

- Two write-protected (WRP) areas can be defined, with page (2 KByte) granularity.

Each area is defined by a start page offset and an end page offset related to the physical Flash base address. These offsets are defined in the WRP address registers: *Flash WRP area A address register (FLASH_WRP1AR)*, *Flash WRP area B address register (FLASH_WRP1BR)*.

The WRP “y” area (x=1 and y=A,B) is defined from the address: *Flash memory Base address + [WRPxy_STRT x 0x800] (included)* to the address: *Flash memory Base address + [(WRPxy_END+1) x 0x800] (excluded)*.

For example, to protect by WRP from the address 0x0801 2800 (included) to the address 0x0801 87FF (included):

- if boot in flash is selected, FLASH_WRP1AR register must be programmed with:
  - WRP1A_STRT = 0x25.
  - WRP1A_END = 0x30.

  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 the FLASH_SR register. This flag is also set for any write access to:

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

Note: When the memory read protection level is selected (RDP level = 1), it is not possible to program or erase Flash 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.

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

Table 25. WRP protection

WRP registers values (x=1 y= A/B)	WRP protection area
WRPxy_STRT = WRPxy_END	Page WRPxy is protected.
WRPxy_STRT > WRPxy_END	No WRP area.
WRPxy_STRT < WRPxy_END	The pages from WRPxy_STRT to WRPxy_END are protected.

4.5.4 Securable memory area

The Securable memory area defines an area of code which can be executed only once at boot, and never again unless a new reset occurs.

The main purpose of the Securable memory area is to protect a specific part of Flash memory against undesired access. This allows implementing software security services such as secure key storage or safe boot. Securable memory area is located in the Main Flash memory. It is dedicated to executing trusted code. When not secured, the Securable memory behaves like the remainder of Main Flash memory. When secured (the SEC_PROT1 bit of the FLASH_CR register set), any attempt to program or erase in a secure memory area generates a write protection error (WRPERR flag is set) and any attempt to read from it generates a read error (RDERR flag is set).

The size of the securable memory area is defined by the SEC_SIZE1[8:0] bitfield of the FLASH_SEC register. It can be modified only in RDP Level 0. Its content is erased upon changing from RDP Level 1 to Level 0, even if it overlaps with PCROP pages.

The securable memory area is defined from the address: Bank base address (included) to the address: Bank base address + (0x800 * SEC_SIZE1) (excluded).

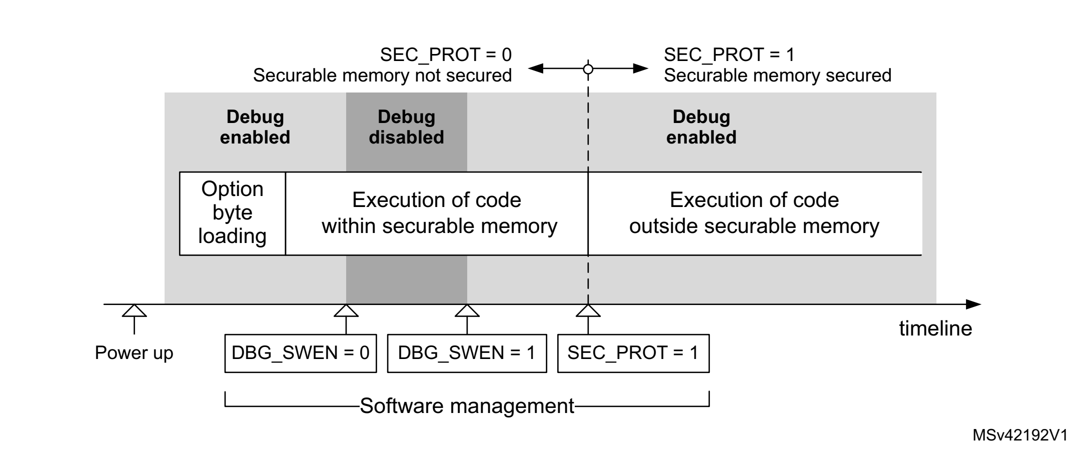
4.5.5 Disabling core debug access

For executing sensitive code or manipulating sensitive data in Securable memory area, the debug access to the core can temporarily be disabled.

In RDP level 2, the debugger is disabled by hardware, , but in other RDP levels, the debugger can be disabled by software using the bit DBG_SWEN in the FLASH_ACR register.

Figure 11 gives an example of managing DBG_SWEN and SEC_PROT bits.

Figure 8. Example of disabling core debug access



4.5.6 Forcing boot from Flash memory

To increase the security and establish a chain of trust, the BOOT_LOCK option bit of the FLASH_SEC1R/FLASH_SEC2R register allows forcing the system to boot from the Main Flash memory regardless the other boot options. It is always possible to set the BOOT_LOCK bit. However, it is possible to reset it only when:

- RDP is set to Level 0, or
- RDP is set to Level 1, while Level 0 is requested and a full mass-erase is performed.

4.6 FLASH interrupts

Table 26. Flash interrupt request

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
Read error	RDERR	Write RDERR=1	RDERRIE
ECC correction	ECCC	Write ECCC=1	ECCCIE

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

4.7 FLASH registers

4.7.1 Flash access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0004 0601

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.	DBG_SWEN	Res.	Res.
													rw		

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_SWEN**: Debug software enable

SW may use this bit to enable/disable the debugger.

0: Debugger disabled

1: Debugger enabled

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **SLEEP_PD**: Flash Power-down mode during Sleep or Low-power sleep mode

This bit determines whether the flash memory is in Power-down mode or Idle mode when the device is in Sleep or Low-power sleep mode.

0: Flash in Idle mode during Sleep and Low-power sleep modes

1: Flash in Power-down mode during Sleep and Low-power sleep modes

Caution: The flash must not be put in power-down while a program or an erase operation is on-going.

Bit 13 **RUN_PD**: Flash Power-down mode during Run or Low-power run mode

This bit is write-protected with FLASH_PDKEYR.

This bit determines whether the flash memory is in Power-down mode or Idle mode when the device is in Run or Low-power run mode. The flash memory can be put in power-down mode only when the code is executed from RAM. The Flash must not be accessed when RUN_PD is set.

0: Flash in Idle mode

1: Flash in Power-down mode

Caution: The flash must not be put in power-down while a program or an erase operation is on-going.

Bit 12 **DCRST**: Data cache reset

0: Data cache is not reset

1: Data cache is reset

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

Bit 11 **ICRST**: Instruction cache reset
0: Instruction cache is not reset
1: Instruction cache is reset
This bit can be written only when the instruction cache is disabled.

Bit 10 **DCEN**: Data cache enable
0: Data cache is disabled
1: Data cache is enabled

Bit 9 **ICEN**: Instruction cache enable
0: Instruction cache is disabled
1: Instruction cache is enabled

Bit 8 **PRFTEN:** Prefetch enable
0: Prefetch disabled
1: Prefetch enabled

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LATENCY[3:0]**: Latency
These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time.
0000: Zero wait state
0001: One wait state
0010: Two wait states
0011: Three wait states
0100: Four wait states
...1111: Fifteen wait states

4.7.2 Flash Power-down key register (FLASH_PDKEYR)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait state, word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDKEYR[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
PDKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDKEYR**: Power-down in Run mode Flash key
The following values must be written consecutively to unlock the RUN_PD bit in FLASH_ACR:
PDKEY1: 0x0415 2637
PDKEY2: 0xFAFB FCFD

4.7.3 Flash key register (FLASH_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait state, word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR[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
KEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEYR**: Flash key

The following values must be written consecutively to unlock the FLACH_CR register allowing flash programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

4.7.4 Flash option key register (FLASH_OPTKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait state, word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEYR[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
OPTKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OPTKEYR**: Option byte key

The following values must be written consecutively to unlock the FLACH_OPTR register allowing option byte programming/erasing operations:
KEY1: 0x0819 2A3B
KEY2: 0x4C5D 6E7F

4.7.5 Flash status register (FLASH_SR)

Address offset: 0x10

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.	BSY
															r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OPTV ERR	RD ERR	Res.	Res.	Res.	Res.	FAST ERR	MISS ERR	PGS ERR	SIZ ERR	PGA ERR	WRP ERR	PROG ERR	Res.	OP ERR	EOP
rc_w1	rc_w1					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

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

Bit 16 **BSY**: Busy

This indicates that a Flash operation is in progress. This is set on the beginning of a Flash operation and reset when the operation finishes or when an error occurs.

Bit 15 **OPTVERR**: Option validity error

Set by hardware when the options read may not be the one configured by the user. If option haven't been properly loaded, OPTVERR is set again after each system reset.
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 (PCROP protection). An interrupt is generated if RDERRIE is set in FLASH_CR.
Cleared by writing 1.

Bits 13:10 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.
Cleared by writing 1.

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

In Fast programming mode, 32 double words must be sent to flash successively, and the new data must be sent to the flash logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.
Cleared by writing 1.

Bit 7 **PGSERR**: Programming sequence error

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. Set also by hardware when PROGERR, SIZERR, PGAERR, WRPERR, MISSERR or FASTERR is set due to a previous programming error.
Cleared by writing 1.

Bit 6 **SIZERR**: Size error

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).

Cleared by writing 1.

Bit 5 **PGAERR**: Programming alignment error

Set by hardware when the data to program cannot be contained in the same 64-bit Flash memory row in case of standard programming, or if there is a change of page during fast programming.

Cleared by writing 1.

Bit 4 **WRPERR**: Write protection error

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.

Cleared by writing 1.

Bit 3 **PROGERR**: Programming error

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

Cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Operation error

Set by hardware when a Flash memory operation (program / erase) completes unsuccessfully.

This bit is set only if error interrupts are enabled (ERRIE = 1).

Cleared by writing ‘1’.

Bit 0 **EOP**: End of operation

Set by hardware when one or more Flash memory operation (programming / erase) has been completed successfully.

This bit is set only if the end of operation interrupts are enabled (EOPIE = 1).

Cleared by writing 1.

4.7.6 Flash control register (FLASH_CR)

Address offset: 0x14

Reset value: 0xC000 0000

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	Res.	SEC_PROT1	OBL_LAUNCH	RD ERRIE	ERR IE	EOP IE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPT STRT	STRT
rs	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.	PNB[7:0]								MER1	PER	PG
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** FLASH_CR Lock

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

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

Bit 29 Reserved, must be kept at reset value

Bit 28 **SEC_PROT1**: Securable memory area protection bit.

This bit is set to lock the access to the Securable memory area. It is set by software when exiting the Securable memory area, and can only be written once.

Bit 27 **OBL_LAUNCH**: Force 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 complete

1: Option byte loading requested

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

This bit enables the interrupt generation when the RDERR bit in the 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 the 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 the 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

This bit triggers an options operation when set.

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

Bit 16 **START**: Start

This bit triggers an erase operation when set. If MER1, MER2 and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition should be forbidden.

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

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:3 **PNB[7:0]**: Page number selection

These bits select the page to erase:

00000000: page 0
00000001: page 1
...
11111111: page 255

Bit 2 **MER1**: Mass erase

This bit triggers the mass erase (all user pages) when set.

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

4.7.7 Flash ECC register (FLASH_ECCR)

Address offset: 0x18

Reset value: 0x0000 0000

Access: no wait state when no Flash memory operation is on going, 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.	ECCC IE	Res.	SYSF_ ECC	Res.	Res.	Res.	ADDR_ECC[18:16]		
rc_w1	rc_w1						rw		r				r	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 (only if ECCC/ECCD are previously cleared). When this bit is set, a NMI is generated.

Cleared by writing 1.

Bit 30 **ECCC**: ECC correction

Set by hardware when one ECC error has been detected and corrected (only if ECCC/ECCC2/ECCD/ECCD2 are previously cleared). An interrupt is generated if ECCCIE is set.

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.

This bit enables the interrupt generation when the ECCC bit in the FLASH_ECCR register is set.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SYSF_ECC**: System Flash ECC fail

This bit indicates that the ECC error correction or double ECC error detection is located in the System Flash.

Bits 21:19 Reserved, must be kept at reset value.

Bits 18:0 **ADDR_ECC**: ECC fail address

This bit indicates which address in the Flash memory is concerned by the ECC error correction or by the double ECC error detection.

4.7.8 Flash option register (FLASH_OPTR)

Address offset: 0x20

Reset value: 0xFXXX XXXX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IRHEN	NRST_MODE [1:0]		nBOOT0	nSWBOOT0	CCMSRAM_RST	SRAM_PE	nBOOT1	PB4_PUPEN	Res.	Res.	WWDG_SW	IWGD_STDBY	IWDG_StOP	IWDG_SW
	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.	nRST_SHDW	nRST_STDBY	nRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
	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 **IRHEN**: Internal reset holder enable bit

0: Internal resets are propagated as simple pulse on NRST pin

1: Internal resets drives NRST pin low until it is seen as low level

Bits 29:28 **NRST_MODE[1:0]**

00: Reserved

01: Reset Input only: a low level on the NRST pin generates system reset, internal RESET not propagated to the NSRT pin

10: GPIO: standard GPIO pad functionality, only internal RESET possible

11: Bidirectional reset: NRST pin configured in reset input/output mode (legacy mode)

Bit 27 **nBOOT0:** nBOOT0 option bit

0: nBOOT0 = 0

1: nBOOT0 = 1

Bit 26 **nSWBOOT0:** Software BOOT0

0: BOOT0 taken from the option bit nBOOT0

1: BOOT0 taken from PB8/BOOT0 pin

Bit 25 **CCMSRAM_RST:** CCM SRAM Erase when system reset

0: CCM SRAM erased when a system reset occurs

1: CCM SRAM is not erased when a system reset occurs

Bit 24 **SRAM_PE:** SRAM1 and CCM SRAM parity check enable

0: SRAM1 and CCM SRAM parity check enable

1: SRAM1 and CCM SRAM parity check disable

Bit 23 **nBOOT1:** Boot configuration

Together with the BOOT0 pin, this bit selects boot mode from the Flash main memory, SRAM1 or the System memory. Refer to *Section 2.6: Boot configuration*.

Bit 22 **PB4_PUPEN:** PB4 pull-up enable

0: USB power delivery dead-battery enabled/ TDI pull-up deactivated

1: USB power delivery dead-battery disabled/ TDI pull-up activated

Note: Only for Category 4 devices (otherwise Reserved)

Bits: 21:20 Reserved, must be kept cleared

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 is frozen in Standby mode

1: Independent watchdog counter is running in Standby mode

Bit 17 **IWDG_STOP:** Independent watchdog counter freeze in Stop mode

0: Independent watchdog counter is frozen in Stop mode

1: Independent watchdog counter is running in Stop mode

Bit 16 **IDWG_SW:** Independent watchdog selection

0: Hardware independent watchdog

1: Software independent watchdog

Bit 15 Reserved, must be kept cleared

Bit 14 **nRST_SHDW**

0: Reset generated when entering the Shutdown mode

1: No reset generated when entering the Shutdown mode

Bit 13 **nRST_STDBY**

0: Reset generated when entering the Standby mode

1: No reset generate when entering the Standby mode

Bit 12 **nRST_STOP**

0: Reset generated when entering the Stop mode

1: No reset generated when entering the Stop mode

Bit 11 Reserved, must be kept cleared

Bits10:8 **BOR_LEV:** BOR reset Level

These bits contain the VDD 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

Bits 7:0 **RDP:** Read protection level

0xAA: Level 0, read protection not active
0xCC: Level 2, chip read protection active
Others: Level 1, memories read protection active

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

4.7.9 Flash PCROP1 Start address register (FLASH_PCROP1SR)

Address offset: 0x24

Reset value: 0xFFFF XXXX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, word 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
PCROP1_STRT[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 cleared

Bits 15:0 **PCROP1_STRT:** PCROP area start offset

PCROP1_STRT contains the first double-word of the PCROP area.

4.7.10 Flash PCROP1 End address register (FLASH_PCROP1ER)

Address offset: 0x28

Reset value: 0xX000 XXXX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, word, half-word access. PCROP_RDP bit can be accessed with byte access.

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
PCROP1_END[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	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 is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:16 Reserved, must be kept cleared

Bits 15:0 **PCROP1_END:** PCROP area end offset

PCROP1_END contains the last double-word of the PCROP area.

4.7.11 Flash WRP area A address register (FLASH_WRP1AR)

Address offset: 0x2C

Reset value: 0x00XX 00XX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, 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.	WRP1A_END[7:0]							
								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.	Res.	Res.	WRP1A_STRT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept cleared

Bits 23:16 **WRP1A_END:** WRP first area “A” end offset
WRP1A_END contains the last page of WRP first area.

Bits 15:8 Reserved, must be kept cleared

Bits 7:0 **WRP1A_STRT:** WRP first area “A” start offset
WRP1A_STRT contains the first page of WRP first area.

4.7.12 Flash WRP area B address register (FLASH_WRP1BR)

Address offset: 0x30

Reset value: 0x00XX 00XX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, 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.	WRP1B_END[7:0]							
								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.	Res.	Res.	WRP1B_STRT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept cleared

Bits 23:16 **WRP1B_END:** WRP second area “B” end offset
WRP1B_END contains the last page of the WRP second area.

Bits 15:8 Reserved, must be kept cleared

Bits 7:0 **WRP1B_STRT:** WRP second area “B” start offset
WRP1B_STRT contains the last page of the WRP second area.

4.7.13 Flash Securable area register (FLASH_SEC1R)

Address offset: 0x70

Reset value: 0xFFFX FXXX

Access: no wait state when no Flash memory operation is on going, 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.	BOOT_LOCK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_SIZE1[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept cleared

Bit 16 **BOOT_LOCK:** used to force boot from user Flash area
0: Boot based on the pad/option bit configuration
1: Boot forced from Main Flash memory

Bits 15:9 Reserved, must be kept cleared

Bit 8:0 **SEC_SIZE1[8:0]**: sets the number of pages used in the Securable area.
Securable area starts at @ 0x0800 0000 and its size is SEC_SIZE1 * page size.
This field can be changed in level0 only.
Any attempt to modify in level1 silently fails, and does not change register value.

4.7.14 FLASH register map

Table 27. 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
0x00	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_SWEN	Res.	Res.	Res.	SLEEP_PD	RUN_PD	DCRST	ICRST	DCEN	ICEN	PRFTEN	Res.	Res.	Res.	Res.	LATENCY [3:0]			
	Reset value														1				0	0	0	0	1	1	0					0	0	0	1
0x04	FLASH_ PDKEYR	PDKEYR[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	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
0x0C	FLASH_OPT KEYR	OPTKEYR[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
0x10	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSY	OPTVERR	RDERR	Res.	Res.	Res.	Res.	FASTERR	MISERR	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value																0	0	0					0	0	0	0	0	0	0		0	0
0x14	FLASH_CR	LOCK	OPTLOCK	Res.	SEC_PROT1	OBL_LAUNCH	RDERRIE	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPTSTRT	STRT	Res.	Res.	Res.	Res.	Res.	PNB[7:0]								MER1	PER	PG
	Reset value	1	1		1	0	0	0	0						0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x18	FLASH_ECCR	ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCCIE	Res.	SYSF_ECC	Res.	Res.	Res.	ADDR_ECC[18:0]																		
	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
0x20	FLASH_OPTR	Res.	Res.	Res.	Res.	nBOOT0	nSWBOOT0	CCMSRAM_RST	SRAM_PE	nBOOT1	PB4_PUPEN	Res.	Res.	WWDG_SW	IWDG_STBY	IWDG_STOP	IWDG_SW	Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	Res.	BOR_ LEV[2:0]			RDP[7:0]							
	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
0x24	FLASH_ PCROP1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_STRT[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x28	FLASH_ PCROP1ER	PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_END[15:0]															
	Reset value	x																X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x2C	FLASH_ WRP1AR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_END[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_STRT[7:0]							
	Reset value									X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X

Table 27. 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
0x30	FLASH_ WRP1BR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_END[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_STRT[7:0]							
	Reset value									X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X
0x70	FLASH_ SEC1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_SIZE1[8:0]								
	Reset value																X								X	X	X	X	X	X	X	X	X

Refer to *Section 2.2 on page 81* for the register boundary addresses.

5 Embedded Flash memory (FLASH) for category 2 devices

5.1 Introduction

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

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

5.2 FLASH main features

- Up to 128 Kbyte of Flash memory (single bank).
- Flash memory read operations with 64 bits data width
- Page erase and mass erase

Flash memory interface features:

- Flash memory read operations
- Flash memory program/erase operations
- Read protection activated by option (RDP)
- 2 Write protection areas selected by option
- Proprietary code read protection areas defined by option
- Securable memory areas defined by option
- Prefetch on ICODE
- Instruction Cache: 32 cache lines of 4 x 64 bits on ICode (1 KB RAM)
- Data Cache: 8 cache lines of 4 x 64 bits on DCode (256B RAM)
- Error Code Correction ECC: 8 bits per 64-bit double-word
  - 8 + 64 = 72 bits, 2 bits detection, 1 bit correction
- Option byte loader
- Low-power mode

5.3 FLASH functional description

5.3.1 Flash memory organization

The Flash memory has the following main features:

- Capacity up to 128 Kbytes (read width of 64-bits)
- 128 KB organized in one single bank for main memory
- Page size of 2 Kbyte
- 72 bits wide data read (64 bits plus 8 ECC bits)
- Page and Mass erase
- Each page is composed of 8 rows of 256 bytes
- An Information block containing:
  - System memory from which the device boots in System memory boot mode. The area is reserved for use by STMicroelectronics and contains the boot loader that is used to reprogram the Flash memory through one of the following interfaces: USART, SPI, I2C, USB. It is programmed by STMicroelectronics when the device is manufactured, and protected against spurious write/erase operations. For further details, please refer to the AN2606 available from *www.st.com*.
  - 1 Kbyte (128 double word) OTP (one-time programmable) bytes 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 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 *Table 28*.

Table 28. Flash module organization (64 bits read width)

Flash area	Flash memory addresses	Size (bytes)	Name
Main memory	0x0800 0000 - 0x0800 07FF	2 K	Page 0
	0x0800 0800 - 0x0800 0FFF	2 K	Page 1
	0x0800 1000 - 0x0800 17FF	2 K	Page 2
	0x0800 1800 - 0x0800 1FFF	2 K	Page 3
	- - - -	- - - -	- - - -
	0x0801 F800 - 0x0801 FFFF	2 K	Page 63
Information block	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	0x1FFF 7800 - 0x1FFF 782F	48	Option bytes

5.3.2 Error code correction (ECC)

Data in Flash memory are 72-bits words: 8 bits are added per double word (64 bits). The ECC mechanism supports:

- One error detection and correction
- Two errors detection

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

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

When an ECC error is detected, the address of the failing double word is saved in ADDR_ECC[20:0] in the FLASH_ECCR register. ADDR_ECC[2:0] are always cleared.

When ECCC or ECCD is set, ADDR_ECC 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 two errors detection 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.

5.3.3 Read access latency

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the *Flash access control register (FLASH_ACR)* according to the frequency of the CPU clock (HCLK) and the internal voltage range of the device V_{CORE}. Refer to *Section 6.1.5: Dynamic voltage scaling management*. *Table 29* shows the correspondence between wait states and CPU clock frequency.

Table 29. Number of wait states according to CPU clock (HCLK) frequency

Wait states (WS) (LATENCY)	HCLK (MHz)		
	V_{CORE} Range 1 boost mode	V_{CORE} Range 1 normal mode	V_{CORE} Range 2
0 WS (1 CPU cycles)	≤ 34	≤ 30	≤ 12
1 WS (2 CPU cycles)	≤ 68	≤ 60	≤ 24
2 WS (3 CPU cycles)	≤ 102	≤ 90	≤ 26
3 WS (4 CPU cycles)	≤ 136	≤ 120	-
4 WS (5 CPU cycles)	≤ 170	≤ 150	-

After reset, the CPU clock frequency is 16 MHz and 1 wait state (WS) is configured in the FLASH_ACR register.

When changing the CPU frequency, the following software sequences must be applied in order to tune the number of wait states needed to access the Flash memory:

Increasing the CPU frequency:

1. Program the new number of wait states to the LATENCY bits in the *Flash access control register (FLASH_ACR)*.
2. Check that the new number of wait states is taken into account to access the Flash memory by reading the FLASH_ACR register.
3. Analyze the change of CPU frequency change caused either by:
   - changing clock source defined by SW bits in RCC_CFGR register
   - or by CPU clock prescaller defined by HPRE bits in RCC_CFGR

If some of above two steps decreases the CPU frequency, firstly perform this step and then the rest. Otherwise modify The CPU clock source by writing the SW bits in the RCC_CFGR register and then (if needed) modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.

4. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register.

Decreasing the CPU frequency:

1. Modify the CPU clock source by writing the SW bits in the RCC_CFGR register.
2. If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.
3. Analyze the change of CPU frequency change caused either by:
   - changing clock source defined by SW bits in RCC_CFGR register
   - or by CPU clock prescaller defined by HPRE bits in RCC_CFGR

If some of above two steps increases the CPU frequency, firstly perform another step and then this step. Otherwise modify The CPU clock source by writing the SW bits in the RCC_CFGR register and then (if needed) modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.

4. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register.
5. Program the new number of wait states to the LATENCY bits in *Flash access control register (FLASH_ACR)*.
6. Check that the new number of wait states is used to access the Flash memory by reading the FLASH_ACR register.

5.3.4 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 FPU processors. It balances the inherent performance advantage of the Arm® Cortex®-M4 with FPU over Flash memory technologies, which normally requires 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 which 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 Flash memory at a CPU frequency up to 170 MHz.

Instruction prefetch

The Cortex®-M4 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.

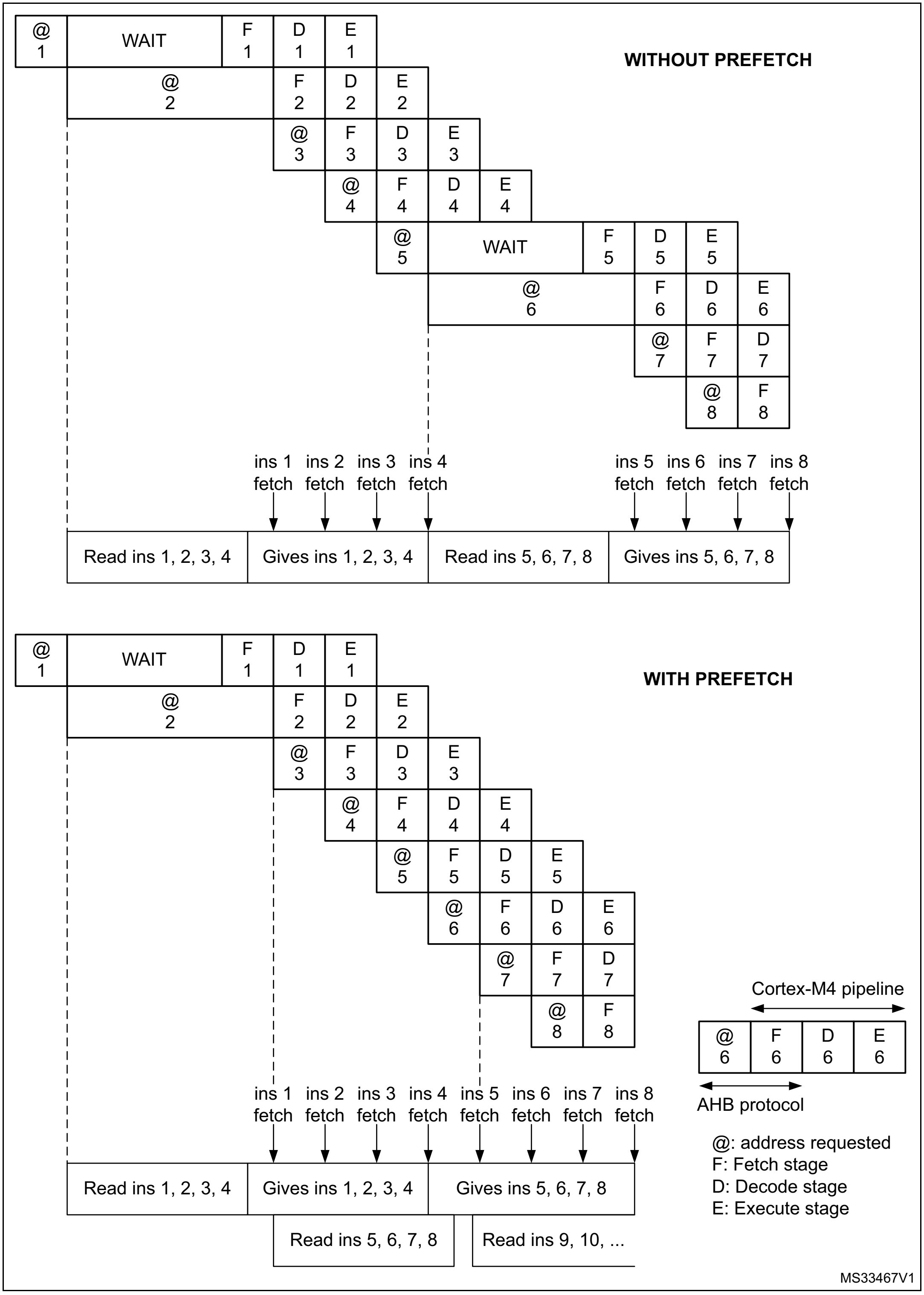
Each Flash memory read operation provides 64 bits from either two instructions of 32 bits or four instructions of 16 bits depending on the launched program. This 64-bits current instruction line is saved in a current buffer, and in case of sequential code, at least two CPU cycles are needed to execute the previous read instruction line.

Prefetch on the 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 the PRFTEN bit in the *Flash access control register (FLASH_ACR)*. This feature is useful if at least one wait state is needed to access the Flash memory.

Figure 9 shows the execution of sequential 16-bit instructions with and without prefetch when 3 WS are needed to access the Flash memory.

Figure 9. Sequential 16-bit instructions execution (64-bit read data width)



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 flash access is performed.

Instruction cache memory (I-Cache)

To limit the time lost due to jumps, it is possible to retain 32 lines of 4 x 64 bits in an instruction cache memory.This feature can be enabled by setting the instruction cache enable (ICEN) bit in the *Flash access control register (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 enable after system reset.

Data cache memory (D-Cache)

Literal pools are fetched from Flash memory through the DCode bus during the execution stage of the CPU pipeline. Each DCode bus read access fetches 64 bits which 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 databus DCode have priority over accesses through the AHB instruction bus ICode.

If some literal pools are frequently used, the data cache memory can be enabled by setting the data cache enable (DCEN) bit in the *Flash access control register (FLASH_ACR)*. This feature works like the instruction cache memory, but the retained data size is limited to 8 rows of 4*64 bits.

The Data cache memory is enable after system reset.

Note: The D-Cache is active only when data is requested by the CPU (not by DMA1 and DMA2).

Data in option bytes block are not cacheable.

5.3.5 Flash program and erase operations

The STM32G4 Series 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 boot loader to load the user application 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 (I/Os, USB, CAN, UART, I^{2}C, SPI, etc.) 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 has to 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.

The Flash erase and programming is only possible in the voltage scaling range 1. The VOS[1:0] bits in the PWR_CR1 must be programmed to 01b.

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 has completed.

Unlocking the Flash memory

After reset, write is not allowed in the *Flash control register (FLASH_CR)* to protect the Flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence is used to unlock this register:

1. Write KEY1 = 0x45670123 in the *Flash key register (FLASH_KEYR)*
2. Write KEY2 = 0xCDEF89AB in the FLASH_KEYR register.

Any wrong sequence locks up the FLASH_CR 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 FLASH_CR register can be locked again by software by setting the LOCK bit in the FLASH_CR register.

Note: The FLASH_CR register cannot be written when the BSY bit in the Flash status register (FLASH_SR) is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

5.3.6 Flash main memory erase sequences

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

Page erase

To erase a page, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH_SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PER bit and select the page to erase (PNB).
4. Set the STRT bit in the FLASH_CR register.
5. Wait for the BSY bit to be cleared in the FLASH_SR register.

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

If the page erase is part of write-protected area (by WRP or PCROP), WRPERR is set and the page erase request is aborted.

Mass erase

To perform a Mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the MER1 bit in the Flash control register (FLASH_CR).
4. Set the STRT bit in the FLACH_CR register.
5. Wait for the BSY bit to be cleared in the Flash status register (FLASH_SR).

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

If the Flasb memory contains a write-protected area (by WRP or PCROP), WRPERR is set and the mass erase request is aborted.

5.3.7 Flash main memory programming sequences

The Flash memory is programmed 72 bits at a time (64 bits + 8 bits ECC).

Programming in a previously programmed address is not allowed except if the data to write is full zero, and any attempt sets PROGERR flag in the *Flash status register (FLASH_SR)*.

It is only possible to program double word (2 x 32-bit data).

- Any attempt to write byte or half-word sets SIZERR flag in the FLASH_SR register.
- Any attempt to write a double word which is not aligned with a double word address sets PGAERR flag in the FLASH_SR register.

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 the BSY bit in the *Flash status register (FLASH_SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PG bit in the *Flash control register (FLASH_CR)*.
4. Perform the data write operation at the desired memory address, inside main memory block or OTP area. Only double word can be programmed.
   - Write a first word in an address aligned with double word
   - Write the second word
5. Wait until the BSY bit is cleared in the FLASH_SR register.
6. Check that EOP flag is set in the FLASH_SR register (meaning that the programming operation has succeed), and clear it by software.
7. Clear the PG bit in the FLASH_SR register if there no more programming request anymore.

Note: When the flash interface has received a good sequence (a double word), programming is automatically launched and BSY bit is set. The internal oscillator HSI16 (16 MHz) is enabled *automatically when PG bit is set, and disabled automatically when PG bit is cleared, except if the HSI16 is previously enabled with HSION in 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.

Fast programming

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

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

The Flash main memory programming sequence in standard mode is as follows:

1. Perform a mass erase. If not, PGSERR is set.
2. Check that no Flash main memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH_SR)*.
3. Check and clear all error programming flag due to a previous programming.
4. Set the FSTPG bit in *Flash control register (FLASH_CR)*.
5. Write the 32 double words to program a row. Only double words can be programmed:
   - Write a first word in an address aligned with double word
   - Write the second word.
6. Wait until the BSY bit is cleared in the FLASH_SR register.
7. Check that EOP flag is set in the FLASH_SR register (meaning that the programming operation has succeed), and clear it by software.
8. Clear the FSTPG bit in the FLASH_SR register if there no more programming request anymore.

Note: If the flash is attempted to be written in Fast programming mode while a read operation is on going, the programming is aborted without any system notification (no error flag is set).

When the Flash interface has received 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 has been programmed or in case of error. The internal oscillator HSI16 (16 MHz) is enabled automatically when FSTPG bit is set, and disabled automatically when FSTPG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

The 32 double word must be written successively. The high voltage is kept on the flash for all the programming. Maximum time between two double words write requests is the time programming (around 2 x 25us). If a second double word arrives after this time programming, fast programming is interrupted and MISSERR is set.

High voltage mustn't exceed 8 ms for a full row between 2 erases. This is guaranteed by the sequence of 32 double words successively written with a clock system greater or equal to 8MHz. An internal time-out counter counts 7ms when Fast programming is set and stops the programming when time-out is over. In this case the FASTERR bit 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

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 is not previously erased (except if the value to program is full zero).
- **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 doesn’t belong to the same double word address.
  - In fast programming: the data to program doesn’t 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: MER1, and PER are not cleared when PG or FSTPG is set.
  - In the fast programming sequence: the Mass erase is not performed before setting FSTPG bit.
  - In the mass erase sequence: PG, FSTPG, and PER are not cleared when MER1 is set.
  - In the page erase sequence: PG, FSTPG, MER1 are not cleared when PER is set.
  - PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR or PGSERR is set due to a previous programming error.
- **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 or in a Securable memory area.
  - Attempt to perform an erase when one page or more is protected by WRP or PCROP.
  - The debug features are connected or the boot is executed from SRAM or from System flash when the read protection (RDP) is set to Level 1.
  - Attempt to modify the option bytes when the read 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 programmation 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 7ms which generates a time-out detection.
- When the fast programming has been 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 the FLASH_SR register:

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 the *Flash status register (FLASH_SR)*, an interrupt is generated and the operation error flag OPERR is set in the FLASH_SR register.

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 requests.

Programming and caches

If a Flash memory write access concerns some data in the data cache, the Flash write access modifies the data in the Flash memory and the data in the cache.

If an erase operation in Flash memory also concerns data in the data or instruction cache, you have to make sure that these data are rewritten before they are accessed during code execution. If this cannot be done safely, it is recommended to flush the caches by setting the DCRST and ICRST bits in the *Flash access control register (FLASH_ACR)*.

Note: The I/D cache should be flushed only when it is disabled (I/DCEN = 0).

5.4 FLASH option bytes

5.4.1 Option bytes description

The option bytes are configured by the end user depending on the application requirements. As a configuration example, the watchdog may be selected in hardware or software mode (refer to *Section 5.4.2: Option bytes programming*).

A double word is split up as follows in the option bytes:

Table 30. Option byte format

63-24	23-16	15 -8	7-0	31-24	23-16	15 -8	7-0
Complemented option byte 3	Complemented option byte 2	Complemented option byte 1	Complemented option byte 0	Option byte 3	Option byte 2	Option byte 1	Option byte 0

The organization of these bytes inside the information block is as shown in *Table 31: Option byte organization*.

The option bytes can be read from the memory locations listed in *Table 31: Option byte organization* or from the Option byte registers:

- *Flash option register (FLASH_OPTR)*
- *Flash PCROP1 Start address register (FLASH_PCROP1SR)*
- *Flash PCROP1 End address register (FLASH_PCROP1ER)*
- *Flash WRP area A address register (FLASH_WRP1AR)*
- *Flash WRP area B address register (FLASH_WRP1BR)*

Table 31. Option byte organization

Address	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
1FFF7800	USER OPT			RDP	USER OPT			RDP
1FFF7808	Unused		Unused and PCROP1_STRT[13:0]		Unused		Unused and PCROP1_STRT[13:0]	
1FFF7810	PCROP_RDP and Unused		Unused and PCROP1_END[13:0]		PCROP_RDP and Unused		Unused and PCROP1_END[13:0]	
1FFF7818	Unused	WRP1A _END [5:0]	Unused	WRP1A _STRT [5:0]	Unused	WRP1A_ END [5:0]	Unused	WRP1A _STRT [5:0]
1FFF7820	Unused	WRP2A _END [5:0]	Unused	WRP2A _STRT [5:0]	Unused	WRP2A_ END [5:0]	Unused	WRP2A _STRT [5:0]
1FFF7828	Unused	BOOT_ LOCK	Unused	SEC_ SIZE1	Unused	BOOT_ LOCK	Unused	SEC_ SIZE1

User and read protection option bytes

Flash memory address: 0x1FFF 7800

ST production value: 0xFFEF F8AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IRH_EN	NRST_MODE		nBOOT0	nSWBOOT0	CCMSRAM_RST	SRAM_PE	nBOOT1	Res.	Res.	Res.	WWDG_SW	IWGD_STDBY	IWDG_STOP	IWDG_SW
	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.	nRST_SHDW	nRST_STDBY	nRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
	r	r	r		r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **IRH_IN**: Internal reset holder for PG10

0: IRH disabled

1: IRH enabled

Bits 29:28 **NRST_MODE**: PG10 pad mode

00: Reset Input/Output

01: Reset Input only

10: GPIO

11: Reset Input/Output

Bit 27 **nBOOT0:** nBOOT0 option bit

0: nBOOT0 = 0

1: nBOOT0 = 1

Bit 26 **nSWBOOT0:** Software BOOT0

0: BOOT0 taken from the option bit nBOOT0

1: BOOT0 taken from PB8/BOOT0 pin

Bit 25 **CCMSRAM_RST:** CCM SRAM erase when system reset

0: CCM SRAM erased when a system reset occurs

1: CCM SRAM is not erased when a system reset occurs

Bit 24 **SRAM_PE:** SRAM1 and CCM SRAM parity check enable

0: SRAM1 and CCM SRAM parity check enable

1: SRAM1 and CCM SRAM parity check disable

Bit 23 **nBOOT1:** Boot configuration

Together with the BOOT0 pin, this bit selects boot mode from the Flash main memory, SRAM1 or the System memory. Refer to *Section 2.6: 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 is frozen in Standby mode

1: Independent watchdog counter is running in Standby mode

Bit 17 **IWDG_STOP:** Independent watchdog counter freeze in Stop mode

0: Independent watchdog counter is frozen in Stop mode

1: Independent watchdog counter is running in Stop mode

Bit 16 **IDWG_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**

0: Reset generated when entering the Shutdown mode
1: No reset generated when entering the Shutdown mode

Bit 13 **nRST_STDBY**

0: Reset generated when entering the Standby mode
1: No reset generate when entering the Standby mode

Bit 12 **nRST_STOP**

0: Reset generated when entering the Stop mode
1: No reset generated when entering the Stop mode

Bit 11 Reserved, must be kept at reset value.

Bits10:8 **BOR_LEV:** BOR reset Level

These bits contain the VDD 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

Bits 7:0 **RDP:** Read protection level

0xAA: Level 0, read protection not active
0xCC: Level 2, chip read protection active
Others: Level 1, memories read protection active

PCROP1 Start address option bytes

Flash memory address: 0x1FFF 7808

ST production value: 0xFFFF FFFF

31	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.	PCROP1_STRT[13:0]													
		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 **PCROP1_STRT:** PCROP area start offset

PCROP1_STRT contains the first double-word of the PCROP area for bank1.

PCROP1 End address option bytes

Flash memory address: 0x1FFF 7810

ST production value: 0x00FF 0000

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.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PCROP1_END[13:0]													
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

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 is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:0 **PCROP1_END:** Bank 1 PCROP area end offset

PCROP1_END contains the last double-word of the PCROP area.

WRP1 Area A address option bytes

Flash memory address: 0x1FFF 7818

ST production value: 0xFF00 FFFF

31	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.	WRP1A_END[5:0]					
										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.	Res.	Res.	WRP1A_STRT[5:0]					
										r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **WRP1A_END:** WRP first area “A” end offset

WRP1A_END contains the last page of WRP first area.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **WRP1A_STRT:** WRP first area “A” start offset

WRP1A_STRT contains the first page of WRP first area.

WRP2 Area A address option bytes

Flash memory address: 0x1FFF 7820

ST production value: 0xFF00 FFFF

31	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.	WRP2A_END[5:0]					
										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.	Res.	Res.	WRP2A_STRT[5:0]					
										r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **WRP2A_END:** WRP second area “B” end offset
WRP1B_END contains the last page of the WRP second area.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **WRP2A_STRT:** WRP second area start offset
WRP1B_STRT contains the last page of the WRP second area.

Securable memory area option bytes

Flash memory address: 0x1FFF7828

ST production value: 0xFF00FF00

31	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.	BOOT_ LOCK
															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.	SEC_SIZE1[6:0]						
									r	r	r	r	r	r	r

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

Bit 16 **BOOT_LOCK** used to force boot from user Flash area
0: Boot based on the pad/option bit configuration
1: Boot forced from Main Flash memory

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

Bits 6:0 **SEC_SIZE1[6:0]:** Securable memory area size
Contains the number of Securable Flash memory pages

5.4.2 Option bytes programming

After reset, the options related bits in the *Flash control register (FLASH_CR)* are write-protected. To run any operation on the option bytes page, the option lock bit OPTLOCK in the *Flash control register (FLASH_CR)* must be cleared. The following sequence is used to unlock this register:

1. Unlock the FLASH_CR with the LOCK clearing sequence (refer to *Unlocking the Flash memory*).
2. Write OPTKEY1 = 0x08192A3B in the *Flash option key register (FLASH_OPTKEYR)*.
3. Write OPTKEY2 = 0x4C5D6E7F in the FLASH_OPTKEYR register.

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 too.

Modifying 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. Check that no Flash memory operation is on going by checking the BSY bit in the *Flash status register (FLASH_SR)*.
2. Clear OPTLOCK option lock bit with the clearing sequence described above.
3. Write the desired options value in the options registers: *Flash option register (FLASH_OPTR)*, *Flash PCROP1 Start address register (FLASH_PCROP1SR)*, *Flash PCROP1 End address register (FLASH_PCROP1ER)*, *Flash WRP area A address register (FLASH_WRP1AR)*, *Flash WRP area B address register (FLASH_WRP1BR)*,
4. Set the Options Start bit OPTSTRT in the *Flash control register (FLASH_CR)*.
5. Wait for the BSY bit to be cleared.

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

Option byte loading

After the BSY bit is cleared, all new options are updated into the flash but they are not applied to the system. They have effect on the system when they are loaded. Option bytes loading (OBL) is performed in two cases:

- when OBL_LAUNCH bit is set in the *Flash control register (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 cannot be read with 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 to check the loading has correctly taken place.

During option byte loading, the options are read by double word with ECC. 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:

- For USR OPT option, the value of mismatch is all options at ‘1’, except for BOR_LEV which 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”

On system reset rising, internal option registers are copied into option registers which can be read and written by software (FLASH_OPTR, FLASH_PCROP1SR, FLASH_PCROP1ER, FLASH_WRP1AR, FLASH_WRP1BR). These registers are also used to modify options. If these registers are not modified by user, they reflects the options states of the system. See *Section : Modifying user options* for more details.

5.5 FLASH memory protection

The Flash main memory can be protected against external accesses with the Read protection (RDP). The pages of the Flash memory can also be protected against unwanted write due to loss of program counter contexts. The write-protection (WRP) granularity is one page (2 KByte). Apart of the flash memory can also be protected against read and write from third parties (PCROP). The PCROP granularity is double word (64-bit).

5.5.1 Read protection (RDP)

The read protection is activated by setting the RDP option byte and then, by applying a system reset to reload the new RDP option byte. The read protection protects to the Flash main memory, the option bytes, the backup registers (RTC_BKPxR in the RTC) and the CCM SRAM.

Note: If the read protection is set while the debugger is still connected through JTAG/SWD, apply a POR (power-on reset) instead of a system reset.

There are three levels of read 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 *Table 32*.

Table 32. Flash memory read protection status

RDP byte value	RDP complement value	Read protection level
0xAA	0x55	Level 0 (production value)
Any value except 0xAA or 0xCC	Any value (not necessarily complementary) except 0x55 and 0x33	Level 1
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 Flash main memory area are possible. The option bytes, the CCM SRAM and the backup registers are also accessible by all operations.

Level 1: Read protection

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

- **User mode:** Code executing in user mode (**Boot Flash**) can access Flash main memory, option bytes, CCM SRAM and backup registers with all operations.
- **Debug, boot RAM and boot loader modes:** In debug mode or when code is running from boot RAM or boot loader, the Flash main memory, the backup registers (RTC_BKPxR in the RTC) and the CCM SRAM are totally inaccessible. In these modes, a read or write access to the Flash generates a bus error and a Hard Fault interrupt.

Caution: In case the Level 1 is configured and no PCROP area is 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 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 Cortex®-M4 debug port, the boot from RAM (boot RAM mode) and the boot from System memory (boot loader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the Flash Main memory. On the contrary, only read operations can be performed on the option bytes.

Option bytes cannot be programmed nor erased. Thus, the level 2 cannot be removed at all: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag WRPERR is set in the Flash_SR register and an interrupt can be generated.

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.

Changing the Read 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 either directly from level 0 or from level 1. Once in level 2, it is no more possible to modify the Read protection level.

When the RDP is reprogrammed to the value 0xAA to move from Level 1 to Level 0, a mass erase of the Flash main memory is performed if PCROP_RDP is set in the *Flash PCROP1 End address register (FLASH_PCROP1ER)*. The backup registers (RTC_BKPxR in the RTC) and the CCM SRAM are also erased. The user options except PCROP protection are set to their previous values copied from FLASH_OPTR, FLASH_WRPxyR (x=1 and y =A or B). PCROP is disable. The OTP area is not affected by mass erase and remains unchanged.

If the bit PCROP_RDP is cleared in the FLASH_PCROP1ER, 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_PCROPxSR and FLASH_PCROPxER registers (x=1).

Note: Full Mass Erase or Partial Mass Erase is performed only when Level 1 is active and Level 0 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 control register.

Figure 10. Changing the read protection (RDP) level

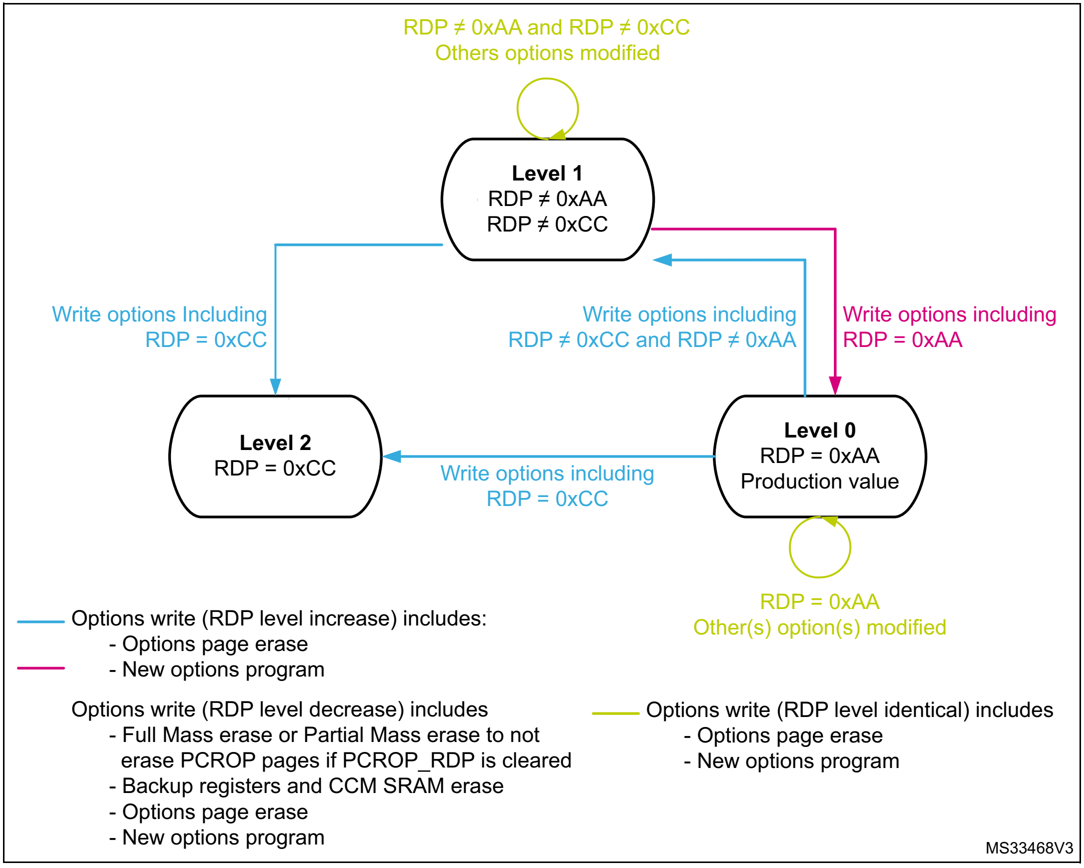


Table 33. Access status versus protection level and execution modes

Area	Protection level	User execution (BootFromFlash)			Debug/ BootFromRam/ BootFromLoader[1]		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No[3]
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory [2]	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A

Table 33. Access status versus protection level and execution modes (continued)

Area	Protection level	User execution (BootFromFlash)			Debug/ BootFromRam/ BootFromLoader[1]		
		Read	Write	Erase	Read	Write	Erase
Option bytes	1	Yes	Yes[3]	Yes	Yes	Yes[3]	Yes
	2	Yes	No	No	N/A	N/A	N/A
OTP	1	Yes	Yes[4]	N/A	No	No	N/A
	2	Yes	Yes[4]	N/A	N/A	N/A	N/A
Backup registers	1	Yes	Yes	N/A	No	No	No[5]
	2	Yes	Yes	N/A	N/A	N/A	N/A
CCM SRAM	1	Yes	Yes	N/A	No	No	No[6]
	2	Yes	Yes	N/A	N/A	N/A	N/A

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 main memory is erased when the RDP option byte is programmed with all level protections disabled (0xAA).
4. OTP can only be written once.
5. The backup registers are erased when RDP changes from level 1 to level 0.
6. The CCM SRAM is erased when RDP changes from level 1 to level 0.

5.5.2 Proprietary code readout protection (PCROP)

Apart 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, as an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. The PCROP area has a double word (64-bit) granularity. An additional option bit (PCROP_RDP) allows to select if the PCROP area is erased or not when the RDP protection is changed from Level 1 to Level 0 (refer to *Changing the Read protection level*).

Each PCROP area is defined by a start page offset and an end page offset related to the physical Flash address. These offsets are defined in the PCROP address registers *Flash PCROP1 Start address register (FLASH_PCROP1SR)*, *Flash PCROP1 End address register (FLASH_PCROP1ER)*.

- The PCROPx (x = 1) area is defined from the address: Flash memory base address + [PCROPx_STRT x 0x8] (included) to the address: Flash memory base address + [(PCROPx_END+1) x 0x8] (excluded). The minimum PCROP area size is two double-words (128 bits).

For example, to protect by PCROP from the address 0x0806 2F80 (included) to the address 0x0807 0004 (included):

- if boot in flash is selected, FLASH_PCROP1SR and FLASH_PCROP1ER registers must be programmed with:
  - PCROP1_STRT = 0xC5F0.
  - PCROP1_END = 0xE000.

Any read access performed through the D-bus to a PCROP protected area triggers 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.

For previous example, due to erase by page, all pages from page 0xC5 to 0xE0 are protected in case of page erase. (All addresses from 0x0806 2800 to 0x0807 07FF can't 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 area, the options programming is launched but PCROP area stays unchanged. On the contrary, it is possible to increase the PCROP area.

When option bit PCROP_RDP is cleared, when the RDP is changing from level 1 to level 0, Full Mass Erase is replaced by Partial Mass Erase in order to keep the PCROP area (refer to *Changing the Read protection level*). In this case, PCROP1_STRT and PCROP1_END are also not erased.

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

Table 34. PCROP protection[1]

PCROPx registers values (x = 1)	PCROP protection area
PCROPx_offset_strt > PCROPx_offset_end	No PCROP area.
PCROPx_offset_strt < PCROPx_offset_end	The area between PCROPx_offset_strt and PCROPx_offset_end is protected. it is possible to write: – PCROPx_offset_strt with a lower value – PCROPx_offset_end with a higher value.

[1] The minimum PCROP area size is 2xdouble words: PCROPx_offset_strt and PCROPx_offset_end.

5.5.3 Write protection (WRP)

The user area in Flash memory can be protected against unwanted write operations. It allows either to specify:

- Two write-protected (WRP) areas can be defined, with page (2 KByte) granularity.

Each area is defined by a start page offset and an end page offset related to the physical Flash base address. These offsets are defined in the WRP address registers: *Flash WRP area A address register (FLASH_WRP1AR)*, *Flash WRP area B address register (FLASH_WRP1BR)*.

The WRP "y" area (x=1 and y=A,B) is defined from the address: *Flash memory Base address + [WRPxy_STRT x 0x800] (included)* to the address: *Flash memory Base address + [(WRPxy_END+1) x 0x800] (excluded)*.

For example, to protect by WRP from the address 0x0801 2800 (included) to the address 0x0801 87FF (included):

- if boot in flash is selected, FLASH_WRP1AR register must be programmed with:
  - WRP1A_STRT = 0x25.
  - WRP1A_END = 0x30.

  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 the FLASH_SR register. This flag is also set for any write access to:

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

Note: When the memory read protection level is selected (RDP level = 1), it is not possible to program or erase Flash 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.

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

Table 35. WRP protection

WRP registers values (x=1 y= A/B)	WRP protection area
WRPxy_STRT = WRPxy_END	Page WRPxy is protected.
WRPxy_STRT > WRPxy_END	No WRP area.
WRPxy_STRT < WRPxy_END	The pages from WRPxy_STRT to WRPxy_END are protected.

5.5.4 Securable memory area

The Securable memory area defines an area of code which can be executed only once at boot, and never again unless a new reset occurs.

The main purpose of the Securable memory area is to protect a specific part of Flash memory against undesired access. This allows implementing software security services such as secure key storage or safe boot. Securable memory area is located in the Main Flash memory. It is dedicated to executing trusted code. When not secured, the Securable memory behaves like the remainder of Main Flash memory. When secured (the SEC_PROT1 bit of the FLASH_CR register set), any attempt to program or erase in a secure memory area generates a write protection error (WRPERR flag is set) and any attempt to read from it generates a read error (RDERR flag is set).

The size of the securable memory area is defined by the SEC_SIZE1[6:0] bitfield of the FLASH_SEC register. It can be modified only in RDP Level 0. Its content is erased upon changing from RDP Level 1 to Level 0, even if it overlaps with PCROP pages.

The securable memory area is defined from the address: Bank base address (included) to the address: Bank base address + (0x800 * SEC_SIZE1) (excluded).

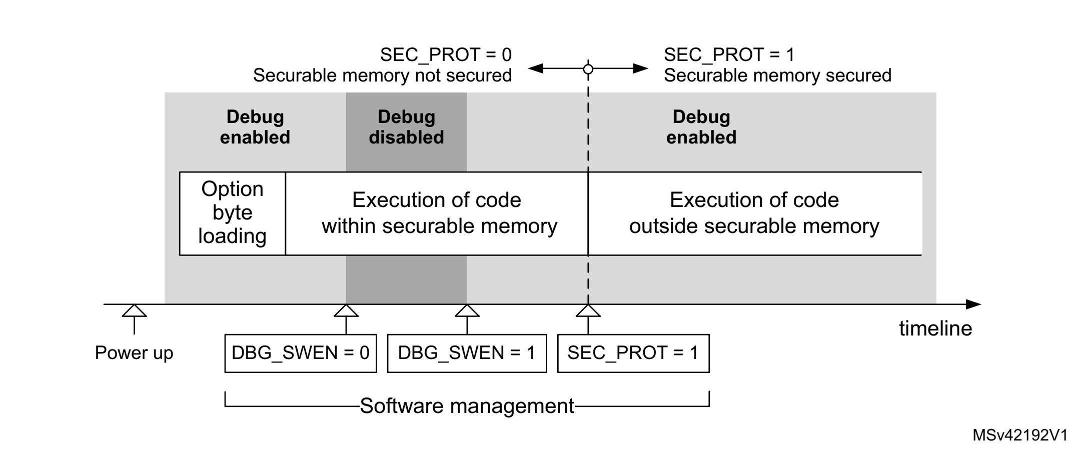
5.5.5 Disabling core debug access

For executing sensitive code or manipulating sensitive data in Securable memory area, the debug access to the core can temporarily be disabled.

In RDP level 2, the debugger is disabled by hardware, , but in other RDP levels, the debugger can be disabled by software using the bit DBG_SWEN in the FLASH_ACR register.

Figure 11 gives an example of managing DBG_SWEN and SEC_PROT bits.

Figure 11. Example of disabling core debug access



5.5.6 Forcing boot from Flash memory

To increase the security and establish a chain of trust, the BOOT_LOCK option bit of the FLASH_SEC1R/FLASH_SEC2R register allows forcing the system to boot from the Main Flash memory regardless the other boot options. It is always possible to set the BOOT_LOCK bit. However, it is possible to reset it only when:

- RDP is set to Level 0, or
- RDP is set to Level 1, while Level 0 is requested and a full mass-erase is performed.

5.6 FLASH interrupts

Table 36. Flash interrupt request

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
Read error	RDERR	Write RDERR=1	RDERRIE
ECC correction	ECCC	Write ECCC=1	ECCCIE

[1] EOP is set only if EOPIE is set.

[2] OPERR is set only if ERRIE is set.

5.7 FLASH registers

5.7.1 Flash access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0004 0601

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.	DBG_ SWEN	Res.	Res.
													rw		

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_SWEN**: Debug software enable

SW may use this bit to enable/disable the debugger.

0: Debugger disabled

1: Debugger enabled

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **SLEEP_PD**: Flash Power-down mode during Sleep or Low-power sleep mode

This bit determines whether the flash memory is in Power-down mode or Idle mode when the device is in Sleep or Low-power sleep mode.

0: Flash in Idle mode during Sleep and Low-power sleep modes

1: Flash in Power-down mode during Sleep and Low-power sleep modes

Caution: The flash must not be put in power-down while a program or an erase operation is on-going.

Bit 13 **RUN_PD**: Flash Power-down mode during Run or Low-power run mode

This bit is write-protected with FLASH_PDKEYR.

This bit determines whether the flash memory is in Power-down mode or Idle mode when the device is in Run or Low-power run mode. The flash memory can be put in power-down mode only when the code is executed from RAM. The Flash must not be accessed when RUN_PD is set.

0: Flash in Idle mode

1: Flash in Power-down mode

Caution: The flash must not be put in power-down while a program or an erase operation is on-going.

Bit 12 **DCRST**: Data cache reset

0: Data cache is not reset

1: Data cache is reset

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

Bit 11 **ICRST**: Instruction cache reset
0: Instruction cache is not reset
1: Instruction cache is reset
This bit can be written only when the instruction cache is disabled.

Bit 10 **DCEN**: Data cache enable
0: Data cache is disabled
1: Data cache is enabled

Bit 9 **ICEN**: Instruction cache enable
0: Instruction cache is disabled
1: Instruction cache is enabled

Bit 8 **PRFTEN:** Prefetch enable
0: Prefetch disabled
1: Prefetch enabled

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LATENCY[3:0]**: Latency
These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time.
0000: Zero wait state
0001: One wait state
0010: Two wait states
0011: Three wait states
0100: Four wait states
...1111: Fifteen wait states

5.7.2 Flash Power-down key register (FLASH_PDKEYR)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait state, word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDKEYR[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
PDKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDKEYR**: Power-down in Run mode Flash key
The following values must be written consecutively to unlock the RUN_PD bit in FLASH_ACR:
PDKEY1: 0x0415 2637
PDKEY2: 0xFAFB FCFD

5.7.3 Flash key register (FLASH_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait state, word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR[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
KEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEYR**: Flash key

The following values must be written consecutively to unlock the FLACH_CR register allowing flash programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

5.7.4 Flash option key register (FLASH_OPTKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait state, word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEYR[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
OPTKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OPTKEYR**: Option byte key

The following values must be written consecutively to unlock the FLACH_OPTR register allowing option byte programming/erasing operations:
KEY1: 0x0819 2A3B
KEY2: 0x4C5D 6E7F

5.7.5 Flash status register (FLASH_SR)

Address offset: 0x10

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.	BSY
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTV ERR	RD ERR	Res.	Res.	Res.	Res.	FAST ERR	MISS ERR	PGS ERR	SIZ ERR	PGA ERR	WRP ERR	PROG ERR	Res.	OP ERR	EOP
rc_w1	rc_w1					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

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

Bit 16 **BSY**: Busy

This indicates that a Flash operation is in progress. This is set on the beginning of a Flash operation and reset when the operation finishes or when an error occurs.

Bit 15 **OPTVERR**: Option validity error

Set by hardware when the options read may not be the one configured by the user. If option haven't been properly loaded, OPTVERR is set again after each system reset.

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 (PCROP protection). An interrupt is generated if RDERRIE is set in FLASH_CR.

Cleared by writing 1.

Bits 13:10 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.

Cleared by writing 1.

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

In Fast programming mode, 32 double words must be sent to flash successively, and the new data must be sent to the flash logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.

Cleared by writing 1.

Bit 7 **PGSERR**: Programming sequence error

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. Set also by hardware when PROGERR, SIZERR, PGAERR, WRPERR, MISSERR or FASTERR is set due to a previous programming error.

Cleared by writing 1.

Bit 6 **SIZERR**: Size error

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).
Cleared by writing 1.

Bit 5 **PGAERR**: Programming alignment error

Set by hardware when the data to program cannot be contained in the same 64-bit Flash memory row in case of standard programming, or if there is a change of page during fast programming.
Cleared by writing 1.

Bit 4 **WRPERR**: Write protection error

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.
Cleared by writing 1.

Bit 3 **PROGERR**: Programming error

Set by hardware when a double-word address to be programmed contains a value different from '0xFFFF FFFF' before programming, except if the data to write is '0x0000 0000'.
Cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Operation error

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

Bit 0 **EOP**: End of operation

Set by hardware when one or more Flash memory operation (programming / erase) has been completed successfully.
This bit is set only if the end of operation interrupts are enabled (EOPIE = 1).
Cleared by writing 1.

5.7.6 Flash control register (FLASH_CR)

Address offset: 0x14

Reset value: 0xC000 0000

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	Res.	SEC_ PROT1	OBL_ LAUNCH	RD ERRIE	ERR IE	EOP IE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPT STRT	STRT
rs	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]							MER1	PER	PG
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** FLASH_CR Lock

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

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

Bit 29 Reserved, must be kept at reset value

Bit 28 **SEC_PROT1**: Securable memory area protection bit.

This bit is set to lock the access to the Securable memory area. It is set by software when exiting the Securable memory area, and can only be written once.

Bit 27 **OBL_LAUNCH**: Force 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 complete

1: Option byte loading requested

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

This bit enables the interrupt generation when the RDERR bit in the 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 the 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 the 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

This bit triggers an options operation when set.

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

Bit 16 **START**: Start

This bit triggers an erase operation when set. If MER1, MER2 and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition should be forbidden.

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

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: Page number selection

These bits select the page to erase:

00000000: page 0
00000001: page 1
...
11111111: page 255

Bit 2 **MER1**: Mass erase

This bit triggers the mass erase (all user pages) when set.

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

5.7.7 Flash ECC register (FLASH_ECCR)

Address offset: 0x18

Reset value: 0x0000 0000

Access: no wait state when no Flash memory operation is on going, 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.	ECCC IE	Res.	SYSF_ ECC	Res.	Res.	Res.	ADDR_ECC[18:16]		
rc_w1	rc_w1						rw		r				r	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 (only if ECCC/ECCD are previously cleared). When this bit is set, a NMI is generated.

Cleared by writing 1.

Bit 30 **ECCC**: ECC correction

Set by hardware when one ECC error has been detected and corrected (only if ECCC/ECCC2/ECCD/ECCD2 are previously cleared). An interrupt is generated if ECCCIE is set.

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.

This bit enables the interrupt generation when the ECCC bit in the FLASH_ECCR register is set.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SYSF_ECC**: System Flash ECC fail

This bit indicates that the ECC error correction or double ECC error detection is located in the System Flash.

Bits 21:19 Reserved, must be kept at reset value.

Bits 18:0 **ADDR_ECC**: ECC fail address

This bit indicates which address in the Flash memory is concerned by the ECC error correction or by the double ECC error detection.

5.7.8 Flash option register (FLASH_OPTR)

Address offset: 0x20

Reset value: 0xFXXX XXXX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IRHEN	NRST_MODE [1:0]		n BOOT0	nSW BOOT0	CCM SRAM_ RST	SRAM _PE	nBOOT 1	Res.	Res.	Res.	WWDG _SW	IWGD_ STDBY	IWDG_ StOP	IWDG_ SW
	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.	nRST_ SHDW	nRST_ STDBY	nRST_ STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
	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 **IRHEN**: Internal reset holder enable bit

0: Internal resets are propagated as simple pulse on NRST pin

1: Internal resets drives NRST pin low until it is seen as low level

Bits 29:28 **NRST_MODE[1:0]**

00: Reserved

01: Reset Input only: a low level on the NRST pin generates system reset, internal RESET not propagated to the NSRT pin

10: GPIO: standard GPIO pad functionality, only internal RESET possible

11: Bidirectional reset: NRST pin configured in reset input/output mode (legacy mode)

Bit 27 **nBOOT0:** nBOOT0 option bit

0: nBOOT0 = 0

1: nBOOT0 = 1

Bit 26 **nSWBOOT0:** Software BOOT0

0: BOOT0 taken from the option bit nBOOT0

1: BOOT0 taken from PB8/BOOT0 pin

Bit 25 **CCMSRAM_RST:** CCM SRAM Erase when system reset

0: CCM SRAM erased when a system reset occurs

1: CCM SRAM is not erased when a system reset occurs

Bit 24 **SRAM_PE:** SRAM1 and CCM SRAM parity check enable

0: SRAM1 and CCM SRAM parity check enable

1: SRAM1 and CCM SRAM parity check disable

Bit 23 **nBOOT1:** Boot configuration

Together with the BOOT0 pin, this bit selects boot mode from the Flash main memory, SRAM1 or the System memory. Refer to *Section 2.6: Boot configuration*.

Bits: 22:20 Reserved, must be kept cleared

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 is frozen in Standby mode

1: Independent watchdog counter is running in Standby mode

Bit 17 **IWDG_STOP:** Independent watchdog counter freeze in Stop mode

0: Independent watchdog counter is frozen in Stop mode

1: Independent watchdog counter is running in Stop mode

Bit 16 **IDWG_SW:** Independent watchdog selection

0: Hardware independent watchdog

1: Software independent watchdog

Bit 15 Reserved, must be kept cleared

Bit 14 **nRST_SHDW**

0: Reset generated when entering the Shutdown mode

1: No reset generated when entering the Shutdown mode

Bit 13 **nRST_STDBY**

0: Reset generated when entering the Standby mode
1: No reset generate when entering the Standby mode

Bit 12 **nRST_STOP**

0: Reset generated when entering the Stop mode
1: No reset generated when entering the Stop mode

Bit 11 Reserved, must be kept cleared

Bits10:8 **BOR_LEV:** BOR reset Level

These bits contain the VDD 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

Bits 7:0 **RDP:** Read protection level

0xAA: Level 0, read protection not active
0xCC: Level 2, chip read protection active
Others: Level 1, memories read protection active

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

5.7.9 Flash PCROP1 Start address register (FLASH_PCROP1SR)

Address offset: 0x24

Reset value: 0xFFFF XXXX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, word 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.	PCROP1_STRT[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept cleared

Bits 14:0 **PCROP1_STRT:** PCROP area start offset

PCROP1_STRT contains the first double-word of the PCROP area.

5.7.10 Flash PCROP1 End address register (FLASH_PCROP1ER)

Address offset: 0x28

Reset value: 0xX000 XXXX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, word, half-word access. PCROP_RDP bit can be accessed with byte access.

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.	PCROP1_END[14:0]														
	rw	rw	rw	rw	rw	rw	rw	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 is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:15 Reserved, must be kept cleared

Bits 14:0 **PCROP1_END:** PCROP area end offset

PCROP1_END contains the last double-word of the PCROP area.

5.7.11 Flash WRP area A address register (FLASH_WRP1AR)

Address offset: 0x2C

Reset value: 0x00XX 00XX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, 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.	WRP1A_END[5:0]					
										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.	WRP1A_STRT[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept cleared

Bits 21:16 **WRP1A_END:** WRP first area “A” end offset

WRP1A_END contains the last page of WRP first area.

Bits 15:6 Reserved, must be kept cleared

Bits 5:0 **WRP1A_STRT:** WRP first area “A” start offset

WRP1A_STRT contains the first page of WRP first area.

5.7.12 Flash WRP area B address register (FLASH_WRP1BR)

Address offset: 0x30

Reset value: 0x00XX 00XX. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going, 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.	WRP1B_END[5:0]					
										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.	WRP1B_STRT[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept cleared

Bits 22:16 **WRP1B_END:** WRP second area “B” end offset

WRP1B_END contains the last page of the WRP second area.

Bits 15:7 Reserved, must be kept cleared

Bits 6:0 **WRP1B_STRT:** WRP second area “B” start offset

WRP1B_STRT contains the last page of the WRP second area.

5.7.13 Flash Securable area register (FLASH_SEC1R)

Address offset: 0x70

Reset value: 0xFFFX FFXX

Access: no wait state when no Flash memory operation is on going, 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.	BOOT_LOCK
															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.	SEC_SIZE1[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept cleared

Bit 16 **BOOT_LOCK** used to force boot from user Flash area
0: Boot based on the pad/option bit configuration
1: Boot forced from Main Flash memory

Bits 15:7 Reserved, must be kept cleared

Bit 6:0 **SEC_SIZE1[6:0]**: sets the number of pages used in the Securable area.
Securable area starts at @ 0x0800 0000 and its size is SEC_SIZE1 * page size.
This field can be changed in level0 only.
Any attempt to modify in level1 silently fails, and does not change register value.

5.7.14 FLASH register map

Table 37. 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
0x00	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_SWEN	Res.	Res.	Res.	SLEEP_PD	RUN_PD	DCRST	ICRST	DCEN	ICEN	PRFTEN	Res.	Res.	Res.	Res.	LATENCY [3:0]			
	Reset value														1				0	0	0	0	1	1	0					0	0	0	1
0x04	FLASH_ PDKEYR	PDKEYR[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	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
0x0C	FLASH_OPT KEYR	OPTKEYR[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
0x10	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSY	OPTVERR	RDERR	Res.	Res.	Res.	Res.	FASTERR	MISERR	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value																0	0	0					0	0	0	0	0	0	0		0	0
0x14	FLASH_CR	LOCK	OPTLOCK	Res.	SEC_PROT1	OBL_LAUNCH	RDERRIE	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPTSTRT	STRT	Res.	Res.	Res.	Res.	Res.	Res.	PNB[6:0]							MER1	PER	PG
	Reset value	1	1		1	0	0	0	0						0	0	0							0	0	0	0	0	0	0	0	0	0
0x18	FLASH_ECCR	ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCCIE	Res.	SYSF_ECC	Res.	Res.	Res.	ADDR_ECC[18:0]																		
	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
0x20	FLASH_OPTR	Res.	Res.	Res.	Res.	nBOOT0	nSWBOOT0	CCMSRAM_RST	SRAM_PE	nBOOT1	Res.	Res.	Res.	WWDG_SW	IWDG_STBY	IWDG_STOP	IWDG_SW	Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	Res.	BOR_ LEV[2:0]			RDP[7:0]							
	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
0x24	FLASH_ PCROP1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_STRT[14:0]														
	Reset value																		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x28	FLASH_ PCROP1ER	PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_END[14:0]														
	Reset value	x																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x2C	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										X	X	X	X	X	X	X										X	X	X	X	X	X	X

Table 37. 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
0x30	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										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x70	FLASH_SEC1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_SIZE1[6:0]						
	Reset value																X										X	X	X	X	X	X	X

Refer to *Section 2.2 on page 81* for the register boundary addresses.

6 Power control (PWR)

6.1 Power supplies

The STM32G4 Series devices require a 1.71 V to 3.6 V operating supply voltage (V_{DD}). Analog peripherals are supplied through independent power domain V_{DDA}.

- V_{DD} = 1.71 V to 3.6 V

  V_{DD} is the external power supply for the I/Os, the internal regulator and the system analog such as reset, power management and internal clocks. It is provided externally through VDD pins.

- V_{DDA} = 1.62 V (ADC/ COMP) / 1.71 V (DAC 1MSPS / DAC 15MSPS) / 2 V (OPAMP) / 2.4 V (VREFBUF)

  V_{DDA} is the external analog power supply for A/D converters, D/A converters, voltage reference buffer, operational amplifiers and comparators. The V_{DDA} voltage level is independent from the V_{DD} voltage. V_{DDA} should be preferably connected to V_{DD} when these peripherals are not used.

  During power up and power down, the following power sequence is required:

  – When V_{DD} is below 1 V, then V_{DDA} supply must remain below V_{DD} + 300 mV

  – When V_{DD} is above 1 V, all power supplies became independent.

  During power down phase, V_{DD} can temporarily become lower then other supplies only if the energy provided to the MCU remains below 1 mJ. this allows external decoupling capacitors to be discarged with different time constants during the power down transient phase.

- V_{BAT} = 1.55 V to 3.6 V

  V_{BAT} is the power supply for RTC, external clock 32 kHz oscillator and backup registers (through power switch) when V_{DD} is not present. VBAT is internally bonded to VDD for small packages without dedicated pin.

- VREF-, VREF+

  V_{REF+} is the input reference voltage for ADCs and DACs. 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 and DAC are not active.

  The internal voltage reference buffer supports three output voltages, which are configured with VRS bit in the VREFBUF_CSR register:

  – V_{REF+} around 2.048 V. This requires V_{DDA} equal to or higher than 2.4 V.

  – V_{REF+} around 2.5 V. This requires V_{DDA} equal to or higher than 2.8 V.

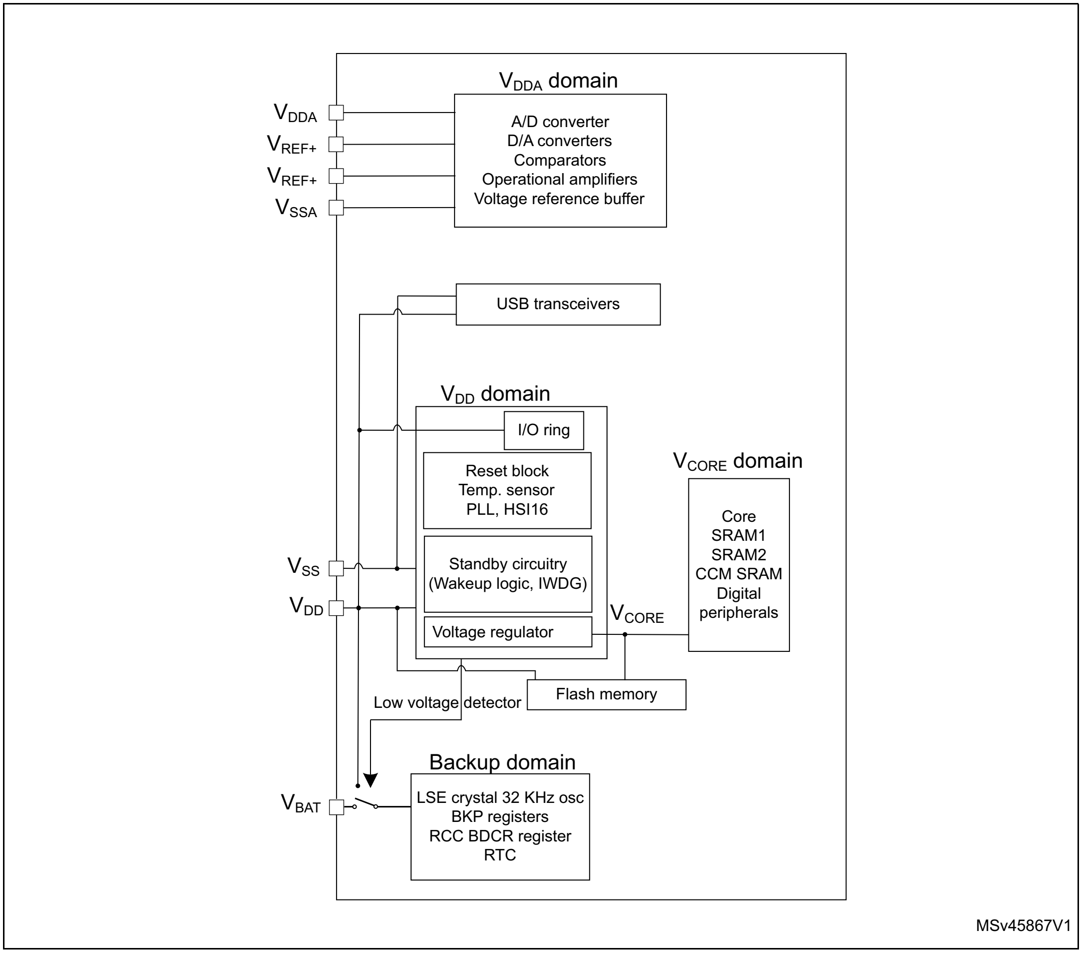
  – V_{REF+} around 2.9 V. This requires V_{DDA} equal to or higher than 3.135 V.

  V_{REF+} pin is not available on all packages. When not available on the package, it is bonded to V_{DDA}. When the V_{REF+} is double-bonded with V_{DDA} in a package, the internal voltage reference buffer (VREFBUF) is not available and must be kept disable (refer to related device datasheet for packages pinout description).

  V_{REF-} is internally double bonded with V_{SSA}.

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, SRAM2 and CCM SRAM. The Flash is supplied by V_{CORE} and V_{DD}.

Figure 12. STM32G4 Series power supply overview



6.1.1 Independent analog peripherals supply

To improve ADC and DAC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply which can be separately filtered and shielded from noise on the PCB.

- The analog peripherals voltage supply input is available on a separate V_{DDA} pin.
- An isolated supply ground connection is provided on V_{SSA} 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, D/A converter, comparators, operational amplifiers, voltage reference buffer).

The V_{DDA} supply can be monitored by the Peripheral Voltage Monitoring, and compared with thresholds. Refer to *Section 6.2.3: 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 and DAC reference voltage

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect to V_{REF+} a separate reference voltage lower than V_{DDA}. V_{REF+} is the highest voltage, represented by the full scale value, for an analog input (ADC) or output (DAC) signal.

V_{REF+} can be provided either by an external reference of by an internal buffered voltage reference (VREFBUF).

The internal buffered voltage reference (VREFBUF) is enabled by setting the ENVR bit in the *Section 23.4.1: VREFBUF control and status register (VREFBUF_CSR)*. The internal buffered voltage reference (VREFBUF) is set to 2.048 V, 2.5 V or 2.9 V according the VRS[1:0] bits setting. The internal buffered voltage reference can also provide the voltage to external components through V_{REF+} pin. Refer to the device datasheet and to *Section 23: Voltage reference buffer (VREFBUF)* for further information.

6.1.2 USB transceivers supply

The USB transceivers are supplied from V_{DD} power supply pin. V_{DD} range for USB usage is from 3.0 V to 3.6 V.

6.1.3 Battery backup domain

To retain the content of the Backup registers and supply the RTC function 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 the RTC unit, the LSE oscillator and the PC13 to PC15 I/Os, allowing the RTC 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 has been 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 may 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 V_{BAT} 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, which can be used as GPIO pins
- PC13, PC14 and PC15, which can be configured by RTC or LSE (refer to *Section 35.3:*

RTC functional description on page 1542)

- PA0/RTC_TAMP2 and PE6/RTC_TAMP3 when they are configured by the RTC 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 has to be limited to 2 MHz with a maximum load of 30 pF and these I/Os must not be used as a 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 or LSE (refer to *Section 35.3: RTC functional description on page 1542*)
- PA0/RTC_TAMP2 and PE6/RTC_TAMP3 when they are configured by the RTC as tamper pins

Backup domain access

After a system reset, the backup domain (RTC registers and backup registers) is protected against possible unwanted write accesses. To enable access to the backup domain, proceed as follows:

1. Enable the power interface clock by setting the PWREN bits in the *Section 7.4.17: APB1 peripheral clock enable register 1 (RCC_APB1ENR1)*
2. Set the DBP bit in the *Power control register 1 (PWR_CR1)* to enable access to the backup domain
3. Select the RTC clock source in the *RTC domain control register (RCC_BDCR)*.
4. Enable the RTC clock by setting the RTCEN [15] bit in the *RTC domain control register (RCC_BDCR)*.

VBAT battery charging

When VDD 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 kOhm resistor or through a 1.5 kOhm resistor depending on the VBRS bit value in the PWR_CR4 register.

The battery charging is enabled by setting VBE bit in the PWR_CR4 register. It is automatically disabled in VBAT mode.

6.1.4 Voltage regulator

Two embedded linear voltage regulators supply all the digital circuitries, except for the Standby circuitry and the backup domain. The main regulator output voltage (V_{CORE}) can be programmed by software to two different power ranges (Range 1 and Range 2) in order to optimize the consumption depending on the system’s maximum operating frequency (refer to *Section 7.2.8: Clock source frequency versus voltage scaling* and to *Section 3.3.3: 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 (MR) or by the low-power regulator (LPR).

- 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 low-power run and low-power sleep modes, the main regulator is off and the low-power regulator (LPR) supplies low power to the V_{CORE} domain, preserving the contents of the registers, SRAM1, SRAM2 and CCM SRAM.
- In Stop 1 modes, the main regulator is off and the low-power regulator (LPR) supplies low power to the V_{CORE} domain, preserving the contents of the registers, SRAM1, SRAM2 and CCM SRAM.
- In Standby mode with SRAM2 content preserved (RRS bit is set in the PWR_CR3 register), the main regulator (MR) is off and the low-power regulator (LPR) provides the supply to SRAM2 only. The core, digital peripherals (except Standby circuitry and backup domain) SRAM1 and CCM SRAM are powered off.
- In Standby mode, both regulators are powered off. The contents of the registers, SRAM1, SRAM2 and CCM SRAM 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, SRAM1, SRAM2 and CCM SRAM is lost, except for the backup domain.

6.1.5 Dynamic voltage scaling management

The dynamic voltage scaling is a power management technique which 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 allows to improve the device performance.

Dynamic voltage scaling to decrease V_{CORE} is known as undervolting. It is performed 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.

In range 1, the main regulator operates in two modes following the R1MODE bit in the PWR_CR5 register:

- Main regulator range 1 normal mode: provides a typical output voltage at 1.2 V. It is used when the system clock frequency is up to 150 MHz. The Flash access time for read access is minimum, write and erase operations are possible.
- Main regulator range 1 boost mode: provides a typical output voltage at 1.28 V. It is used when the system clock frequency is up to 170 MHz. The Flash access time for read access is minimum, write and erase operations are possible. To optimize the power consumption it is recommended to select the range1 boost mode when the system clock frequency is greater than 150 MHz. See *Table 38*.

Table 38. Range 1 boost mode configuration

System frequency	SYSCLK ≤ 150 MHz	SYSCLK ≤ 170 MHz
R1MODE bit configuration	1	0

- 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 26 MHz.The Flash access time for a read access is increased as compared to Range 1; write and erase operations are not possible.

Voltage scaling is selected through the VOS bit in the *Section 6.4.1: Power control register 1 (PWR_CR1)* register.

The sequence to go from Range 1 (Normal/Boost) to Range 2 is:

1. In case of switching from Range 1 boost mode to Range 2, the system clock must be divided by 2 using the AHB prescaler before switching to a lower system frequency for at least 1us and then reconfigure the AHB prescaler.
2. Reduce the system frequency to a value lower than 26 MHz.
3. Adjust number of wait states according new frequency target in Range 2 (LATENCY bits in the FLASH_ACR).
4. Program the VOS bits to “10” in the PWR_CR1 register.

The sequence to go from Range 2 to Range 1 (normal/boost mode) is:

1. Program the VOS bits to “01” in the PWR_CR1 register.
2. Wait until the VOSF flag is cleared in the PWR_SR2 register.
3. Adjust number of wait states according new frequency target in Range 1 (LATENCY bits in the FLASH_ACR).
4. Increase the system frequency by following below procedure:

- If the system frequency is 26 MHz < SYSCLK ≤ 150 MHz:
  - Select the Range 1 normal mode by setting R1MODE bit in the PWR_CR5 register.
  - Configure and switch to PLL for a new system frequency.
- If the system frequency is SYSCLK > 150 MHz:
  - The system clock must be divided by 2 using the AHB prescaler before switching to a higher system frequency.
  - Select the Range 1 boost mode by clearing the R1MODE bit is in the PWR_CR5 register.
  - Configure and switch to PLL for a new system frequency.
  - Wait for at least 1us and then reconfigure the AHB prescaler to get the needed HCLK clock frequency.

The sequence to switch from Range1 normal mode to Range1 boost mode is:

1. The system clock must be divided by 2 using the AHB prescaler before switching to a higher system frequency.
2. Clear the R1MODE bit is in the PWR_CR5 register.
3. Adjust the number of wait states according to the new frequency target in range1 boost mode
4. Configure and switch to new system frequency.
5. Wait for at least 1us and then reconfigure the AHB prescaler to get the needed HCLK clock frequency.

The sequence to switch from Range1 boost mode to Range1 normal mode is:

1. Set the R1MODE bit is in the PWR_CR5 register.
2. Adjust the number of wait states according new frequency target in Range1 default mode.
3. Configure and switch to new system frequency.

6.2 Power supply supervisor

6.2.1 Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR)

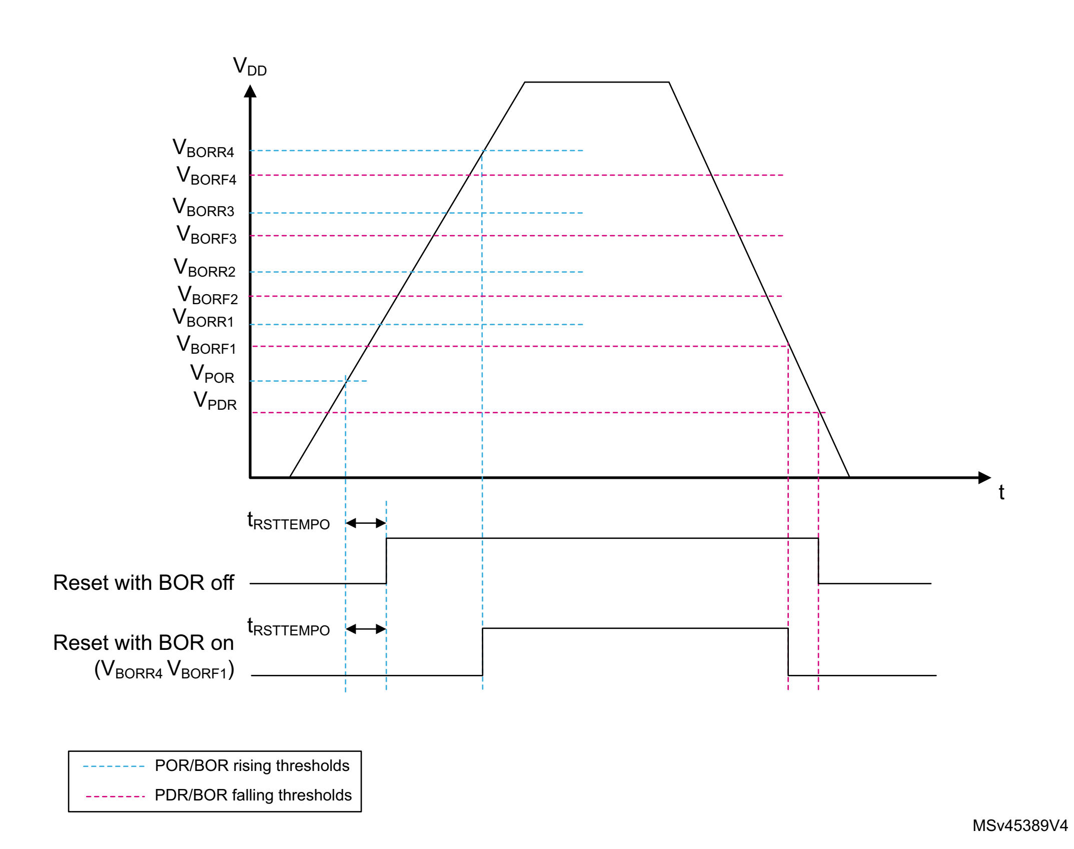
The device has an integrated power-on reset (POR) / power-down reset (PDR), coupled with a brown-out reset (BOR) circuitry. The BOR is active in all power modes except Shutdown mode, and cannot be disabled.

Five BOR thresholds can be selected through option bytes.

During power-on, the 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 brown-out reset thresholds, refer to the electrical characteristics section in the datasheet.

Figure 13. Brown-out reset waveform



1. The reset temporization $t_{RSTTEMPO}$ is present only for the BOR lowest threshold (V_{BOR0}).

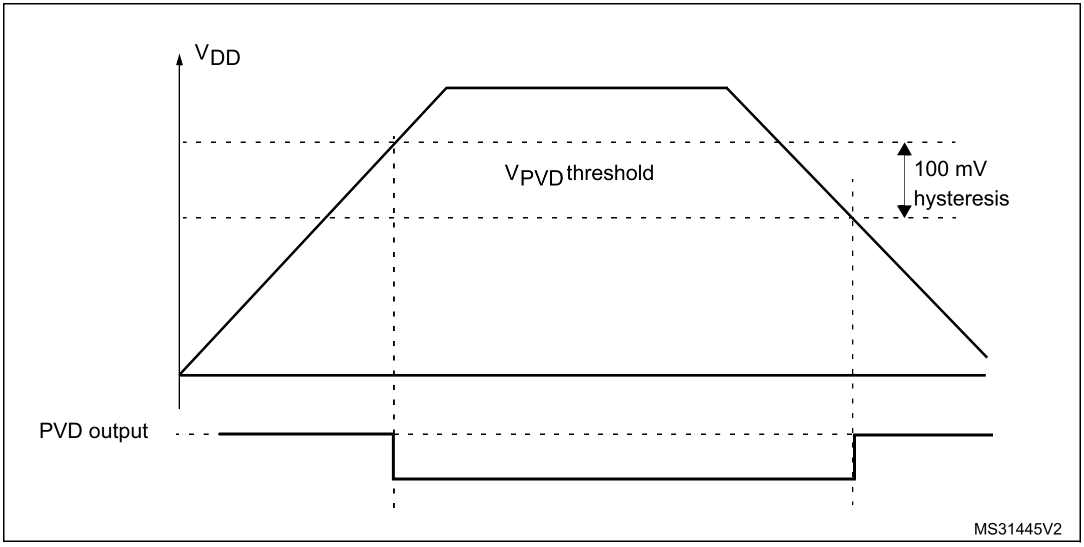
6.2.2 Programmable voltage detector (PVD)

You can use the PVD to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the *Power control register 2 (PWR_CR2)*.

The 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} is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when V_{DD} drops below the PVD threshold and/or when V_{DD} rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example, the service routine could perform emergency shutdown tasks.

Figure 14. PVD thresholds



6.2.3 Peripheral Voltage Monitoring (PVM)

Only V_{DD} is monitored by default, as it is the only supply required for all system-related functions. The V_{DDA} can be independent from V_{DD} and can be monitored with two peripheral voltage monitoring (PVM).

Each of the four PVMx (x=1, 2) is a comparator between a fixed threshold VPVMx and the V_{DDA} power supply. PVMOx flags indicate if the independent power supply is higher or lower than the PVMx threshold: 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 and Stop 1 modes, and the PVM interrupt can wake up from the Stop mode.

Table 39. PVM features

PVM	Power supply	PVM threshold	EXTI line
PVM1	V_{DDA}	V_{PVM1} (around 1.65 V)	40
PVM2	V_{DDA}	V_{PVM2} (around 1.8 V)	41

The independent analog 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 this dedicated supply.

- If V_{DDA} is shorted externally to V_{DD}, the application should assume it 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 supply is present or not.

6.3 Low-power modes

By default, the microcontroller is in Run mode after a system or a power Reset. Several low-power modes are available to save power when the CPU does not need to be kept running, for example when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources.

The device features seven low-power modes:

- Sleep mode: CPU clock off, all peripherals including Cortex®-M4 with FPU core peripherals such as NVIC, SysTick, etc. can run and wake up the CPU when an interrupt or an event occurs. Refer to *Section 6.3.4: Sleep mode*.
- Low-power run mode: This mode is achieved when the CPU clock frequency is reduced below 2 MHz. The code is executed from the SRAM or the Flash memory. The regulator is in low-power mode to minimize the regulator's operating current. Refer to *Section 6.3.2: Low-power run mode (LP run)*.
- Low-power sleep mode: This mode is entered from the Low-power run mode: Cortex®-M4 with FPU is off. Refer to *Section 6.3.5: Low-power sleep mode (LP sleep)*.
- Stop 0 and Stop 1 modes: SRAM and all registers content are retained. All clocks in the V_{CORE} domain are stopped, the PLL, the HSI16 and the HSE are disabled. The LSI and the LSE can be kept running.

  The RTC and TAMP can remain active (Stop mode with RTC, Stop mode without RTC).

  Some peripherals with the wakeup capability can enable the HSI16 RC during the Stop mode to detect their wakeup condition.

  In Stop 0 mode, the main regulator remain ON, which allows the fastest wakeup time but with higher consumption. The active peripherals and the wakeup sources are the same as in Stop 1 mode.

  The system clock, when exiting from Stop 0 or Stop 1 mode, is the HSI16 clock. If the device is configured to wake up in low-power run mode, the HPRE bits in RCC_CFGR register must be configured prior to entering Stop mode to provide a frequency not greater than 2 MHz.

  Refer to *Section 6.3.6: Stop 0 mode* for details on Stop 0 mode.
- Standby mode: V_{CORE} domain is powered off. However, it is possible to preserve the SRAM contents:
  - Standby mode with SRAM2 retention when the bit RRS is set in PWR_CR3 register. In this case, SRAM2 is supplied by the low-power regulator.
  - Standby mode when the bit RRS is cleared in PWR_CR3 register. In this case the main regulator and the low-power regulator are powered off.

  All clocks in the V_{CORE} domain are stopped, the PLL, the HSI16 and the HSE oscillator are disabled. The LSI and the LSE can be kept running.

  The RTC can remain active (Standby mode with RTC, Standby mode without RTC).

  The system clock, when exiting Standby modes, is the HSI16 oscillator clock.

  Refer to *Section 6.3.8: Standby mode*.
- Shutdown mode: V_{CORE} domain is powered off. All clocks in the V_{CORE} domain are stopped, the PLL, the HSI16, the LSI and the HSE are disabled. The LSE can be kept running. The system clock, when exiting the Shutdown mode, is HSI16 oscillator clock. In this mode, the supply voltage monitoring is disabled and the product behavior is not guaranteed in case of a power voltage drop. Refer to *Section 6.3.9: Shutdown mode*.

In addition, the power consumption in Run mode can be reduced by one of the following means:

- Slowing down the system clocks
- Gating the clocks to the APB and AHB peripherals when they are unused.

Figure 15. Low-power modes possible transitions

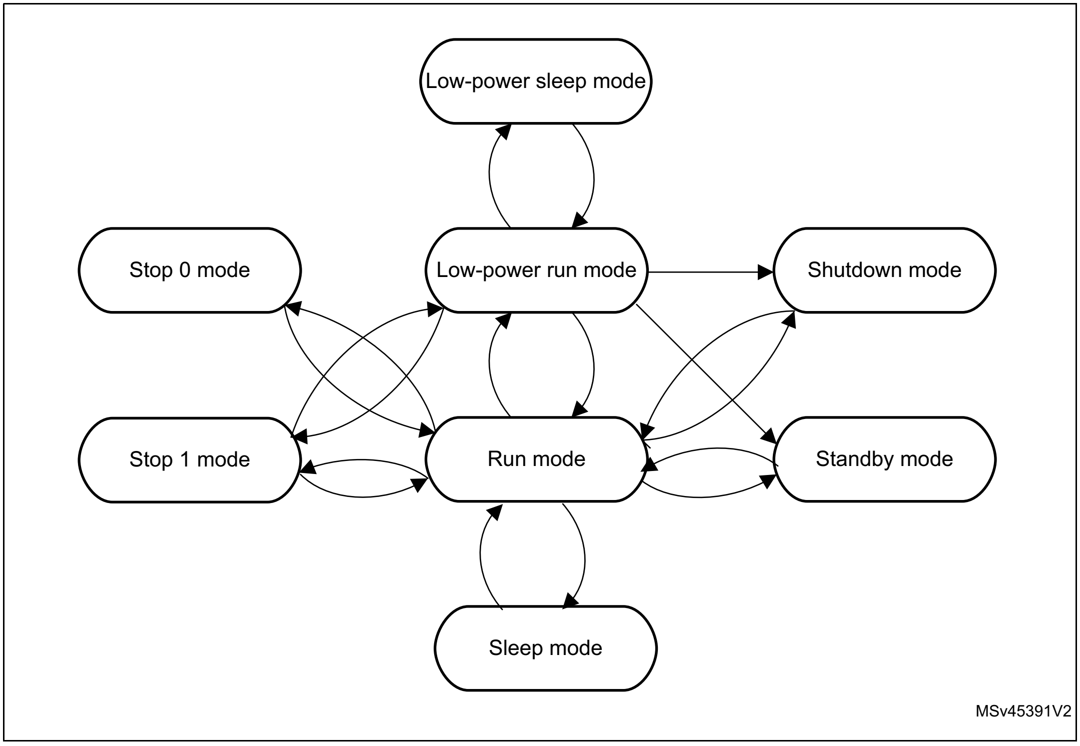


Table 40. Low-power mode summary

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				
Low-power run	Set LPR bit	Clear LPR bit	Same as Low-power run clock	None	OFF	
Low-power sleep	Set LPR bit + WFI or Return from ISR	Any interrupt	Same as before entering Low-power sleep mode	CPU clock OFF no effect on other clocks or analog clock sources		
	Set LPR bit + WFE	Wakeup event				
Stop 0	LPMS="000" + SLEEPDEEP bit + WFI or Return from ISR or WFE	Any EXTI line (configured in the EXTI registers) Specific peripherals events	HSI16	All clocks OFF except LSI and LSE	ON	
Stop 1	LPMS="001" + SLEEPDEEP bit + WFI or Return from ISR or WFE				OFF	
Standby with SRAM2	LPMS="011"+ Set RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, TAMP event, external reset on NRST pin, IWDG reset				
Standby	LPMS="011" + Clear RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE					OFF
Shutdown	LPMS="1--" + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, TAMP event, external reset on NRST pin		All clocks OFF except LSE		

1. Refer to *Table 41: Functionalities depending on the working mode*.

Table 41. Functionalities depending on the working mode[1]

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Standby		Shutdown		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
CPU	Y	-	Y	-	-	-	-	-	-	-	-
Flash memory	O[2]	O[2]	O[2]	O[2]	-	-	-	-	-	-	-
SRAM1	Y	Y[3]	Y	Y[3]	Y	-	-	-	-	-	-
SRAM2	Y	Y[3]	Y	Y[3]	Y	-	O[4]	-	-	-	-
CCM SRAM	Y	Y[3]	Y	Y[3]	Y	-	-	-	-	-	-
FSMC	O	O	O	O	-	-	-	-	-	-	-
QUADSPI	O	O	O	O	-	-	-	-	-	-	-
Backup Registers	Y	Y	Y	Y	Y	-	Y	-	Y	-	Y
Brown-out reset (BOR)	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-
Programmable Voltage Detector (PVD)	O	O	O	O	O	O	-	-	-	-	-
Peripheral Voltage Monitor (PVM)	O	O	O	O	O	O	-	-	-	-	-
DMA	O	O	O	O	-	-	-	-	-	-	-
Oscillator HSI16	O	O	O	O	[5]	-	-	-	-	-	-
Oscillator HSI48	O	O	-	-	-	-	-	-	-	-	-
High Speed External (HSE)	O	O	O	O	-	-	-	-	-	-	-
Low Speed Internal (LSI)	O	O	O	O	O	-	O	-	-	-	-
Low Speed External (LSE)	O	O	O	O	O	-	O	-	O	-	O
Clock Security System (CSS)	O	O	O	O	-	-	-	-	-	-	-
Clock Security System on LSE	O	O	O	O	O	O	O	O	-	-	-
RTC / Auto wakeup	O	O	O	O	O	O	O	O	O	O	O
Number of RTC Tamper pins	3	3	3	3	3	O	3	O	3	O	3
USB	O[8]	O[8]	-	-	-	O	-	-	-	-	-
USARTx (x=1,2,3,4,5)	O	O	O	O	O[6]	O[6]	-	-	-	-	-
Low-power UART (LPUART1)	O	O	O	O	O[6]	O[6]	-	-	-	-	-
I2Cx (x=1,2,3,4)	O	O	O	O	O[7]	O[7]	-	-	-	-	-

Table 41. Functionalities depending on the working mode[1] (continued)

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Standby		Shutdown		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
SPIx (1,2,3,4)	O	O	O	O	-	-	-	-	-	-	-
FDCANx (1,2,3)	O	O	O	O	-	-	-	-	-	-	-
SAI1	O	O	O	O	-	-	-	-	-	-	-
ADCx (x=1,2,3,4,5)	O	O	O	O	-	-	-	-	-	-	-
DACx (x=1,2,3,4)	O	O	O	O	O	-	-	-	-	-	-
VREFBUF	O	O	O	O	O	-	-	-	-	-	-
OPAMPx (x=1,2,3,4,5,6)	O	O	O	O	O	-	-	-	-	-	-
COMPx (x=1,2,3,4,5,6,7)	O	O	O	O	O	O	-	-	-	-	-
Temperature sensor	O	O	O	O	-	-	-	-	-	-	-
Timers (TIMx)	O	O	O	O	-	-	-	-	-	-	-
High resolution timer 1 (HRTIM1)	O	O	O	O	-	-	-	-	-	-	-
Low-power timer 1 (LPTIM1)	O	O	O	O	O	O	-	-	-	-	-
Independent watchdog (IWDG)	O	O	O	O	O	O	O	O	-	-	-
Window watchdog (WWDG)	O	O	O	O	-	-	-	-	-	-	-
SysTick timer	O	O	O	O	-	-	-	-	-	-	-
Random number generator (RNG)	O[8]	O[8]	-	-	-	-	-	-	-	-	-
AES hardware accelerator	O	O	O	O	-	-	-	-	-	-	-
CRC calculation unit	O	O	O	O	-	-	-	-	-	-	-
GPIOs	O	O	O	O	O	O	[9]	5 pins [10]	[11]	5 pins [10]	-
Filter Mathematical Accelerator (FMAC)	O	O	O	O	-	-	-	-	-	-	-
CORDIC co-processor (CORDIC)	O	O	O	O	-	-	-	-	-	-	-

1. Legend: Y = Yes (Enable). O = Optional (Disable by default. Can be enabled by software). - = Not available , wakeup highlighted in gray.
2. The Flash can be configured in power-down mode. By default, it is not in power-down mode.
3. The SRAM clock can be gated on or off.
4. SRAM2 content is preserved when the bit RRS is set in PWR_CR3 register.

5. 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 which requested it. HSI16 is automatically put off when the peripheral does not need it anymore.
6. UART and LPUART reception is functional in Stop mode, and generates a wakeup interrupt on Start, address match or received frame event.
7. I2C address detection is functional in Stop mode, and generates a wakeup interrupt in case of address match.
8. Voltage scaling Range 1 only.
9. I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
10. The I/Os with wakeup from Standby/Shutdown capability are: PA0, PC13, PE6, PA2, PC5.
11. 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.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop 0, Stop1, Standby or Shutdown mode while the debug features are used. This is due to the fact that the Cortex®-M4 with FPU 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 47.16.1: Debug support for low-power modes.*

6.3.1 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 *Section 7.4.3: Clock configuration register (RCC_CFGR)*.

Peripheral clock gating

In Run mode, the HCLK and PCLK for individual peripherals and memories can be stopped at any time to reduce the power consumption.

To further reduce the power consumption in Sleep mode, 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 bit in the RCC_AHBxSMENR and RCC_APBxSMENR registers.

6.3.2 Low-power run mode (LP run)

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 CPU frequency should not exceed 2 MHz.

Please refer to the product datasheet for more details on voltage regulator and peripherals operating conditions.

I/O states in Low-power run mode

In Low-power run mode, all I/O pins keep the same state as in Run mode.

Entering the Low-power run mode

To enter the Low-power run mode, proceed as follows:

1. Optional: Jump into the SRAM and power-down the Flash by setting the RUN_PD bit in the *Flash access control register (FLASH_ACR)*.
2. Decrease the CPU clock frequency below 2 MHz.
3. Force the regulator in low-power mode by setting the LPR bit in the PWR_CR1 register.

Refer to *Table 42: Low-power run* on how to enter the Low-power run mode.

Exiting the Low-power run mode

To exit the Low-power run mode, proceed as follows:

1. Force the regulator in main mode by clearing the LPR bit in the PWR_CR1 register.
2. Wait until REGLPF bit is cleared in the PWR_SR2 register.
3. Increase the CPU clock frequency.

Refer to *Table 42: Low-power run* on how to exit the Low-power run mode.

Table 42. Low-power run

Low-power run mode	Description
Mode entry	Decrease the CPU clock frequency below 2 MHz LPR = 1
Mode exit	LPR = 0 Wait until REGLPF = 0 Increase the CPU clock frequency
Wakeup latency	Regulator wakeup time from low-power mode

6.3.3 Low power modes

Entering low power mode

Low power modes are entered by the MCU by executing the WFI (Wait For Interrupt), or WFE (Wait for Event) instructions, or when the SLEEPONEXIT bit in the Cortex®-M4 with FPU System Control register is set on Return from ISR.
Entering Low-power mode through WFI or WFE is executed only if no interrupt is pending or no event is pending.

Exiting low power mode

From Sleep modes, and Stop modes the MCU exit low power mode depending on the way the low power mode was entered:

- 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 MCU exits the low power mode as soon as an event occurs. The wakeup event can be generated either

by:

- NVIC IRQ interrupt.

  - When SEVONPEND = 0 in the Cortex®-M4 with FPU System Control register. By enabling an interrupt in the peripheral control register and in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

  Only NVIC interrupts with sufficient priority wakeup and interrupt the MCU.

  - When SEVONPEND = 1 in the Cortex®-M4 with FPU System Control register.

  By enabling an interrupt in the peripheral control register and optionally in the NVIC. When the MCU 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) have to be cleared.

  All NVIC interrupts wakeup the MCU, even the disabled ones. Only enabled NVIC interrupts with sufficient priority wakeup and interrupt the MCU.

- Event

  Configuring a 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 modes, and Shutdown modes the MCU exit low power mode through an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC event occurs (see *Figure 528: RTC block diagrams*).
After waking up from Standby or Shutdown mode, program execution restarts in the same way as after a Reset (boot pin sampling, option bytes loading, reset vector is fetched, etc.).

6.3.4 Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

Entering the Sleep mode

The Sleep mode is entered according *Section : Entering low power mode*, when the SLEEPDEEP bit in the Cortex®-M4 with FPU System Control register is clear.

Refer to *Table 43: Sleep* for details on how to enter the Sleep mode.

Exiting the Sleep mode

The Sleep mode is exit according Section : Exiting low power mode.

Refer to *Table 43: Sleep* for more details on how to exit the Sleep mode.

Table 43. Sleep

Sleep-now 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®-M4 with FPU System Control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex®-M4 with FPU System Control register.
Mode exit	If WFI or return from ISR was used for entry Interrupt: refer to *Table 97: STM32G4 Series vector table* If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to *Section 15.3.2: Wakeup event management* If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to *Table 97: STM32G4 Series vector table* or Wakeup event: refer to *Section 15.3.2: Wakeup event management*
Wakeup latency	None

6.3.5 Low-power sleep mode (LP sleep)

Please refer to the product datasheet for more details on voltage regulator and peripherals operating conditions.

I/O states in Low-power sleep mode

In Low-power sleep mode, all I/O pins keep the same state as in Run mode.

Entering the Low-power sleep mode

The Low-power sleep mode is entered from low-power run mode according *Section : Entering low power mode*, when the SLEEPDEEP bit in the Cortex®-M4 with FPU System Control register is clear.

Refer to *Table 44: Low-power sleep* for details on how to enter the Low-power sleep mode.

Exiting the Low-power sleep mode

The low-power Sleep mode is exit according *Section : Exiting low power mode*. When exiting the Low-power sleep mode by issuing an interrupt or an event, the MCU is in Low-power run mode.

Refer to *Table 44: Low-power sleep* for details on how to exit the Low-power sleep mode.

Table 44. Low-power sleep

Low-power sleep-now mode	Description
Mode entry	Low-power sleep mode is entered from the Low-power run 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®-M4 with FPU System Control register.
	Low-power sleep mode is entered from the Low-power run mode. On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex®-M4 with FPU System Control register.
Mode exit	If WFI or Return from ISR was used for entry Interrupt: refer to *Table 97: STM32G4 Series vector table* If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to *Section 15.3.2: Wakeup event management* If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to *Table 97: STM32G4 Series vector table* Wakeup event: refer to *Section 15.3.2: Wakeup event management* After exiting the Low-power sleep mode, the MCU is in Low-power run mode.
Wakeup latency	None

6.3.6 Stop 0 mode

The Stop 0 mode is based on the Cortex®-M4 with FPU deepsleep 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; the PLL, the HSI16 and the HSE oscillators are disabled. Some peripherals with the wakeup capability (I2Cx (x=1,2,3,4), U(S)ARTx(x=1,2...5) and LPUART) can switch on the HSI16 to receive a frame, and switch off the 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, CCM SRAM 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.

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.

Entering the Stop 0 mode

The Stop 0 mode is entered according *Section : Entering low power mode*, when the SLEEPDEEP bit in the Cortex®-M4 with FPU System Control register is set.

Refer to *Table 45: Stop 0 mode* for details on how to enter the Stop 0 mode.

If Flash memory programming is ongoing, the Stop 0 mode entry is delayed until the memory access is finished.

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 42.3: IWDG functional description*.
- real-time clock (RTC): this is configured by the RTCEN bit in the *RTC domain control register (RCC_BDCR)*
- Internal RC oscillator (LSI): this is configured by the LSION bit in the *Control/status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the *RTC domain control register (RCC_BDCR)*.

Several peripherals can be used in Stop 0 mode and can add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock: LPTIM1, I2Cx (x=1,2,3,4) U(S)ARTx(x=1,2...5), LPUART.

The DACx (x=1,2,3,4), the OPAMPs and the comparators can be used in Stop 0 mode, the PVM and the PVD as well. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx (x=1,2,3,4,5), temperature sensor and VREFBUF buffer can consume power during the Stop 0 mode, unless they are disabled before entering this mode.

Exiting the Stop 0 mode

The Stop 0 mode is exit according *Section : Entering low power mode*.

Refer to *Table 45: Stop 0 mode* for details on how to exit Stop 0 mode.

When exiting Stop 0 mode by issuing an interrupt or a wakeup event, the HSI16 oscillator is selected as system clock. If the device is configured to wake up in Low-power run mode, the HPRE bits in RCC_CFGR register must be configured prior to entering Stop 0 mode to provide a frequency not greater than 2 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 although 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_CR1) or in Low-power run mode if the bit LPR is set in the Power control register 1 (PWR_CR1).

Table 45. Stop 0 mode

Stop 0 mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “000” in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = “000” in PWR_CR1
	Note: To enter Stop 0 mode, all EXTI Line pending bits (in Pending register 1 (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 97: STM32G4 Series vector table*. If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to *Section 15.3.2: Wakeup event management*. 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 97: STM32G4 Series vector table*. Wakeup event: refer to *Section 15.3.2: Wakeup event management*
Wakeup latency	Longest wakeup time between: HSI16 wakeup time and Flash wakeup time from Stop 0 mode.

6.3.7 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 Low-power run mode.

Refer to *Table 46: Stop 1 mode* for details on how to enter and exit Stop 1 mode.

Table 46. Stop 1 mode

Stop 1 mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “001” in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = “001” in PWR_CR1
	Note: To enter Stop 1 mode, all EXTI Line pending bits (in Section 15.5.6: Pending register 1 (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 97: STM32G4 Series vector table*. If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to *Section 15.3.2: Wakeup event management*. 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 97: STM32G4 Series vector table*. Wakeup event: refer to *Section 15.3.2: Wakeup event management*
Wakeup latency	Longest wakeup time between: HSI16 wakeup time and regulator wakeup time from Low-power mode + Flash wakeup time from Stop 1 mode.

6.3.8 Standby mode

The Standby mode allows to achieve the lowest power consumption with BOR. It is based on the Cortex®-M4 with FPU deepsleep mode, with the voltage regulators disabled (except when SRAM2 content is preserved). The PLL, the HSI16, and the HSE oscillators are also switched off.

SRAM1 and register contents are lost except for registers in the Backup domain and Standby circuitry (see *Figure 12*). SRAM2 content can be preserved if the bit RRS is set in the PWR_CR3 register. In this case the Low-power regulator is ON and provides the supply to SRAM2 only.

The BOR is always available in Standby mode. The consumption is increased when thresholds higher than V_{BOR0} are used.

I/O states in Standby mode

In the Standby mode, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x=A,B,C,D,E,F,G)), or with a pull-down (refer to PWR_PDCRx registers (x=A,B,C,D,E,F,G)), or can be kept in analog state.

The RTC outputs on PC13 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. 5 wakeup pins (WKUPx, x=1,2...5) and the 3 RTC tampers are available.

Entering Standby mode

The Standby mode is entered according *Section : Entering low power mode*, when the SLEEPDEEP bit in the Cortex®-M4 with FPU System Control register is set.

Refer to *Table 47: Standby mode* for details on how to enter Standby mode.

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 42.3: IWDG functional description* in *Section 42: Independent watchdog (IWDG)*.
- 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 LSION 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)

Exiting Standby mode

The Standby mode is exit according *Section : Entering low power mode*. The SBF status flag in the *Power control register 3 (PWR_CR3)* indicates that the MCU was in Standby mode. All registers are reset after wakeup from Standby except for *Power control register 3 (PWR_CR3)*.

Refer to *Table 47: Standby mode* for more details on how to exit Standby mode.

Table 47. Standby mode

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “011” 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®-M4 with FPU System Control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = “011” in PWR_CR1 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, tamper or timestamp flags) is cleared
Mode exit	WKUPx pin edge, RTC event, external Reset in NRST pin, IWDG Reset, BOR reset
Wakeup latency	Reset phase

6.3.9 Shutdown mode

The Shutdown mode allows to achieve the lowest power consumption. It is based on the deepsleep mode, with the voltage regulator disabled. The V_{CORE} domain is consequently powered off. The PLL, the HSI16, the LSI and the HSE oscillators are also switched off.

SRAM1, SRAM2, CCM SRAM and register contents are lost except for registers in the Backup domain. The BOR is not available in Shutdown mode. No power voltage monitoring is possible in this mode, therefore the switch to Backup domain is not supported.

I/O states in Shutdown mode

In the Shutdown mode, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x=A,B,C,D,E,F,G), or with a pull-down (refer to PWR_PDCRx registers (x=A,B,C,D,E,F,G)), 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 outputs on PC13 are functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. 5 wakeup pins (WKUPx, x=1,2...5) and the 3 RTC tampers are available.

Entering Shutdown mode

The Shutdown mode is entered according *Entering low power mode*, when the SLEEPDEEP bit in the Cortex®-M4 with FPU System Control register is set.

Refer to *Table 48: Shutdown mode* for details on how to enter Shutdown mode.

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 VDD 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)

Exiting Shutdown mode

The Shutdown mode is exit according *Section : Exiting low power mode*. 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 *Table 48: Shutdown mode* for more details on how to exit Shutdown mode.

Table 48. Shutdown mode

Shutdown mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “1XX” 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®-M4 with FPU System Control register – SLEEPONEXT = 1 – No interrupt is pending – LPMS = “1XX” in PWR_CR1 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, tamper or timestamp flags) is cleared
Mode exit	WKUPx pin edge, RTC event, external Reset in NRST pin
Wakeup latency	Reset phase

6.3.10 Auto-wakeup from low-power mode

The RTC can be used to wakeup 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 or 1) or Standby mode at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the *RTC 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 17 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 17.

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.

6.4 PWR registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

6.4.1 Power control register 1 (PWR_CR1)

Address offset: 0x00

Reset value: 0x0000 0200. This register is reset after wakeup from 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.	LPR	Res.	Res.	Res.	VOS[1:0]		DBP	Res.	Res.	Res.	Res.	Res.	LPMS[2:0]		
	rw				rw	rw	rw						rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **LPR**: Low-power run

When this bit is set, the regulator is switched from main mode (MR) to low-power mode (LPR).

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:9 **VOS**: Voltage scaling range selection

00: Cannot be written (forbidden by hardware)
01: Range 1
10: Range 2
11: 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 7:3 Reserved, must be kept at reset value.

Bits 2:0 **LPMS[2:0]**: Low-power mode selection

These bits select the low-power mode entered when CPU enters the deepsleep mode.

000: Stop 0 mode
001: Stop 1 mode
010: Reserved
011: Standby mode
1xx: Shutdown mode

Note: In Standby mode, SRAM2 can be preserved or not, depending on RRS bit configuration in PWR_CR3.

6.4.2 Power control register 2 (PWR_CR2)

Address offset: 0x04

Reset value: 0x0000 0000. This register is reset when exiting 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.	PVMEN2	PVMEN1	Res.	Res.	PLS[2:0]			PVDE
								rw	rw			rw	rw	rw	rw

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

Bit 7 **PVMEN2**: Peripheral voltage monitoring 4 enable: V_{DDA} vs. DAC 1MSPS /DAC 15MSPS min voltage.

0: PVM2 (V_{DDA} monitoring vs. 1.8 V threshold) disable.
1: PVM2 (V_{DDA} monitoring vs. 1.8 V threshold) enable.

Bit 6 **PVMEN1**: Peripheral voltage monitoring 3 enable: V_{DDA} vs. ADC/COMP min voltage 1.62V

0: PVM1 (V_{DDA} monitoring vs. 1.62V threshold) disable.
1: PVM1 (V_{DDA} monitoring vs. 1.62V 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 PVD falling threshold:
000: V_{PVD0} PVD threshold 0
001: V_{PVD1} PVD threshold 1
010: V_{PVD2} PVD threshold 2
011: V_{PVD3} PVD threshold 3
100: V_{PVD4} PVD threshold 4
101: V_{PVD5} PVD threshold 5
110: V_{PVD6} PVD threshold 6
111: External input analog voltage PVD_IN (compared internally to V_{REFINT})

Note: These bits are write-protected when the PVD_LOCK bit is set in the SYSCFG_CFGR2 register. The protection can be reset only by a system reset.

Bit 0 **PVDE**: Programmable voltage detector enable

0: Programmable voltage detector disable.
1: Programmable voltage detector enable.

Note: This bit is write-protected when the PVD_LOCK bit is set in the SYSCFG_CFGR2 register. *The protection can be reset only by a system reset.*

6.4.3 Power control register 3 (PWR_CR3)

Address offset: 0x08

Reset value: 0x0000 8000. This register is not reset when exiting Standby modes and with the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

31	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	UCPD1 _DBDIS	UCPD1_ STDBY	Res.	Res.	APC	Res.	RRS	Res.	Res.	Res.	EWUP 5	EWUP 4	EWUP 3	EWUP 2	EWUP 1
rw	rw	rw			rw		rw				rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EIWUL**: Enable internal wakeup line

0: Internal wakeup line disable.

1: Internal wakeup line enable.

Bit 14 **UCPD1_DBDIS**: USB Type-C and Power Delivery Dead Battery disable.

After exiting reset, the USB Type-C “dead battery” behavior is enabled, which may have a pull-down effect on CC1 and CC2 pins. It is recommended to disable it in all cases, either to stop this pull-down or to hand over control to the UCPD1 (which should therefore be initialized before doing the disable).

0: Enable USB Type-C dead battery pull-down behavior on UCPD1_CC1 and UCPD1_CC2 pins.

1: Disable USB Type-C dead battery pull-down behavior on UCPD1_CC1 and UCPD1_CC2 pins.

Bit 13 **UCPD1_STDBY:** UCPD1_STDBY USB Type-C and Power Delivery standby mode.

0: Write ‘0’ immediately after standby exit when using UCPD1, (and before writing any UCPD1 registers).

1: Write ‘1’ just before entering standby when using UCPD1.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **APC**: Apply pull-up and pull-down configuration

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 Reserved, must be kept at reset value.

Bit 8 **RRS**: SRAM2 retention in Standby mode

0: SRAM2 is powered off in Standby mode (SRAM2 content is lost).

1: SRAM2 is powered by the low-power regulator in Standby mode (SRAM2 content is kept).

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **EWUP5**: Enable Wakeup pin WKUP5

When this bit is set, the external wakeup pin WKUP5 is enabled and triggers a wakeup from Standby or Shutdown event when a rising or a falling edge occurs.The active edge is configured via the WP5 bit in the PWR_CR4 register.

Bit 3 **EWUP4**: Enable Wakeup pin WKUP4

When this bit is set, the external wakeup pin WKUP4 is enabled and triggers a wakeup from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP4 bit in the PWR_CR4 register.

Bit 2 **EWUP3**: Enable Wakeup pin WKUP3

When this bit is set, the external wakeup pin WKUP3 is enabled and triggers a wakeup from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP3 bit in the PWR_CR4 register.

Bit 1 **EWUP2**: Enable Wakeup pin WKUP2

When this bit is set, the external wakeup pin WKUP2 is enabled and triggers a wakeup from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP2 bit in the PWR_CR4 register.

Bit 0 **EWUP1**: Enable Wakeup pin WKUP1

When this bit is set, the external wakeup pin WKUP1 is enabled and triggers a wakeup from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP1 bit in the PWR_CR4 register.

6.4.4 Power control register 4 (PWR_CR4)

Address offset: 0x0C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

31	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.	VBRS	VBE	Res.	Res.	Res.	WP5	WP4	WP3	WP2	WP1
						rw	rw				rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **VBRS**: V_{BAT} battery charging resistor selection

0: Charge V_{BAT} through a 5 kOhms resistor

1: Charge V_{BAT} through a 1.5 kOhms resistor

Bit 8 **VBE**: V_{BAT} battery charging enable

0: V_{BAT} battery charging disable

1: V_{BAT} battery charging enable

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **WP5**: Wakeup pin WKUP5 polarity

This bit defines the polarity used for an event detection on external wake-up pin, WKUP5

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 3 **WP4**: Wakeup pin WKUP4 polarity

This bit defines the polarity used for an event detection on external wake-up pin, WKUP4

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

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)

6.4.5 Power status register 1 (PWR_SR1)

Address offset: 0x10

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with the PWRRST bit in the RCC_APB1RSTR1 register.

Access: 2 additional APB cycles are needed to read this register vs. a standard APB read.

31	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.	Res.	Res.	Res.	SBF	Res.	Res.	Res.	WUF5	WUF4	WUF3	WUF2	WUF1
r							r				r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **WUFI**: Wakeup flag internal

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 14:9 Reserved, must be kept at reset value.

Bit 8 **SBF**: Standby flag

This bit is set by hardware when the device enters the Standby mode and is cleared by setting the CSBF bit in the PWR_SCR register, or by a power-on reset. It is not cleared by the system reset.
0: The device did not enter the Standby mode
1: The device entered the Standby mode

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **WUF5**: Wakeup flag 5

This bit is set when a wakeup event is detected on wakeup pin, WKUP5. It is cleared by writing ‘1’ in the CWUF5 bit of the PWR_SCR register.

Bit 3 **WUF4**: Wakeup flag 4

This bit is set when a wakeup event is detected on wakeup pin,WKUP4. It is cleared by writing ‘1’ in the CWUF4 bit of the PWR_SCR register.

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_SCR register.

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_SCR register.

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_SCR register.

6.4.6 Power status register 2 (PWR_SR2)

Address offset: 0x14

Reset value: 0x0000 0000. This register is partially reset when exiting Standby/Shutdown modes.

31	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**
PVMO2	PVMO1	Res.	Res.	PVDO	VOSF	REGLPF	REGLPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r			r	r	r	r								

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PVMO2**: Peripheral voltage monitoring output: V_{DDA} vs. 1.8 V

0: V_{DDA} voltage is above PVM2 threshold (around 1.8 V).

1: V_{DDA} voltage is below PVM2 threshold (around 1.8 V).

Note: PVMO2 is cleared when PVM2 is disabled (PVME = 0). After enabling PVM2, the PVM2 output is valid after the PVM2 wakeup time.

Bit 14 **PVMO1**: Peripheral voltage monitoring output: V_{DDA} vs. 1.62 V

0: V_{DDA} voltage is above PVM1 threshold (around 1.62 V).

1: V_{DDA} voltage is below PVM1 threshold (around 1.62 V).

Note: PVMO1 is cleared when PVM1 is disabled (PVME = 0). After enabling PVM1, the PVM1 output is valid after the PVM1 wakeup time.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **PVDO**: Programmable voltage detector output

0: V_{DD} is above the selected PVD threshold

1: V_{DD} is 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_CR1 register.

0: The regulator is ready in the selected voltage range

1: The regulator output voltage is changing to the required voltage level

Bit 9 **REGLPF**: Low-power regulator flag

This bit is set by hardware when the MCU is in Low-power run mode. When the MCU exits the Low-power run mode, this bit remains at 1 until the regulator is ready in main mode. A polling on this bit must be done before increasing the product frequency.

This bit is cleared by hardware when the regulator is ready.

0: The regulator is ready in main mode (MR)

1: The regulator is in low-power mode (LPR)

Bit 8 **REGLPS**: Low-power regulator started

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, the wakeup from Standby mode time may be increased.

0: The low-power regulator is not ready

1: The low-power regulator is ready

Bit 7 **FLASH_RDY:** Flash ready flag

This bit is set by hardware to indicate when the Flash memory is ready to be accessed after wakeup from power-down. To place the Flash memory in power-down, set either FPD_LPRUN, FPD_LPSLP or FPD_STP bits.

0: Flash memory in power-down

1: Flash memory ready to be accessed

Note: If the system boots from SRAM, the user application must wait till FLASH_RDY bit is set, prior to jumping to Flash memory.

Bits 6:0 Reserved, must be kept at reset value.

6.4.7 Power status clear register (PWR_SCR)

Address offset: 0x18

Reset value: 0x0000 0000.

Access: 3 additional APB cycles are needed to write this register vs. a standard APB write.

31	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.	CSBF	Res.	Res.	Res.	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
							w				w	w	w	w	w

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **CSBF**: Clear standby flag

Setting this bit clears the SBF flag in the PWR_SR1 register.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CWUF5**: Clear wakeup flag 5

Setting this bit clears the WUF5 flag in the PWR_SR1 register.

Bit 3 **CWUF4**: Clear wakeup flag 4

Setting this bit clears the WUF4 flag in the PWR_SR1 register.

Bit 2 **CWUF3**: Clear wakeup flag 3
Setting this bit clears the WUF3 flag in the PWR_SR1 register.

Bit 1 **CWUF2**: Clear wakeup flag 2
Setting this bit clears the WUF2 flag in the PWR_SR1 register.

Bit 0 **CWUF1**: Clear wakeup flag 1
Setting this bit clears the WUF1 flag in the PWR_SR1 register.

6.4.8 Power Port A pull-up control register (PWR_PUCRA)

Address offset: 0x20.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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	Res.	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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PU15**: Port A pull-up bit 15
When set, this bit activates the pull-up on PA[15] when APC bit is set in PWR_CR3 register. The pull-up is not activated if the corresponding PD15 bit is also set.

Bit 14 Reserved, must be kept at reset value.

Bits 13:0 **PUy**: Port A pull-up bit y (y=0..13)
When set, this bit activates the pull-up on PA[y] when APC bit is set in PWR_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

6.4.9 Power Port A pull-down control register (PWR_PDCRA)

Address offset: 0x24.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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.	PD14	Res.	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

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **PD14**: Port A pull-down bit 14

When set, this bit activates the pull-down on PA[14] when APC bit is set in PWR_CR3 register.

Bit 13 Reserved, must be kept at reset value.

Bits 12:0 **PDy**: Port A pull-down bit y (y=0..12)

When set, this bit activates the pull-down on PA[y] when APC bit is set in PWR_CR3 register.

6.4.10 Power Port B pull-up control register (PWR_PUCRB)

Address offset: 0x28.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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 **PUy**: Port B pull-up bit y (y=0..15)

When set, this bit activates the pull-up on PB[y] when APC bit is set in PWR_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

6.4.11 Power Port B pull-down control register (PWR_PDCRB)

Address offset: 0x2C.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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	Res.	PD3	PD2	PD1	PD0
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 15:5 **PDy**: Port B pull-down bit y (y=5..15)
When set, this bit activates the pull-down on PB[y] when APC bit is set in PWR_CR3 register.

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **PDy**: Port B pull-down bit y (y=0..3)
When set, this bit activates the pull-down on PB[y] when APC bit is set in PWR_CR3 register.

6.4.12 Power Port C pull-up control register (PWR_PUCRC)

Address offset: 0x30.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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 **PUy**: Port C pull-up bit y (y=0..15)
When set, this bit activates the pull-up on PC[y] when APC bit is set in PWR_CR3 register.
The pull-up is not activated if the corresponding PDy bit is also set.

6.4.13 Power Port C pull-down control register (PWR_PDCRC)

Address offset: 0x34.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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 **PDy**: Port C pull-down bit y (y=0..15)
When set, this bit activates the pull-down on PC[y] when APC bit is set in PWR_CR3 register.

6.4.14 Power Port D pull-up control register (PWR_PUCRD)

Address offset: 0x38.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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 **PUy**: Port D pull-up bit y (y=0..15)
When set, this bit activates the pull-up on PD[y] when APC bit is set in PWR_CR3 register.
The pull-up is not activated if the corresponding PDy bit is also set.

6.4.15 Power Port D pull-down control register (PWR_PDCRD)

Address offset: 0x3C.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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 **PDy**: Port D pull-down bit y (y=0..15)
When set, this bit activates the pull-down on PD[y] when APC bit is set in PWR_CR3 register.

6.4.16 Power Port E pull-up control register (PWR_PUCRE)

Address offset: 0x40.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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 **PUy**: Port E pull-up bit y (y=0..15)
When set, this bit activates the pull-up on PE[y] when APC bit is set in PWR_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

6.4.17 Power Port E pull-down control register (PWR_PDCRE)

Address offset: 0x44.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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 **PDy**: Port E pull-down bit y (y=0..15)
When set, this bit activates the pull-down on PE[y] when APC bit is set in PWR_CR3 register.

6.4.18 Power Port F pull-up control register (PWR_PUCRF)

Address offset: 0x48.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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 **PUy**: Port F pull-up bit y (y=0..15)
When set, this bit activates the pull-up on PF[y] when APC bit is set in PWR_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

6.4.19 Power Port F pull-down control register (PWR_PDCRF)

Address offset: 0x4C.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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 **PDy**: Port F pull-down bit y (y=0..15)
When set, this bit activates the pull-down on PF[y] when APC bit is set in PWR_CR3 register.

6.4.20 Power Port G pull-up control register (PWR_PUCRG)

Address offset: 0x50.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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.	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PUy**: Port G pull-up bit y (y=0..10)
When set, this bit activates the pull-up on PG[y] when APC bit is set in PWR_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

6.4.21 Power Port G pull-down control register (PWR_PDCRG)

Address offset: 0x54.

Reset value: 0x0000 0000. 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 vs. a standard APB access (3 for a write and 2 for a read).

31	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.	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PDy**: Port G pull-down bit y (y=0..10)

When set, this bit activates the pull-down on PG[y] when APC bit is set in PWR_CR3 register.

6.4.22 Power control register (PWR_CR5)

Address offset: 0x80.

Reset value: 0x0000 0100. This register is reset only by power on reset.

31	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.	R1MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

Bits 31:9 Reserved, must be kept at reset value.

Bits 15:0 **R1MODE**: Main regular range 1 mode

This bit is only valid for the main regulator in range 1 and has no effect on range 2. It is recommended to reset this bit when the system frequency is greater than 150 MHz. Refer to *Table 38: Range 1 boost mode configuration*.

0: Main regulator in range 1 boost mode.

1: Main regulator in range 1 normal mode.

Bits 7:0 Reserved, must be kept at reset value.

6.4.23 PWR register map and reset value table

Table 49. PWR 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	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.	Res.	Res.	FPD_STOP	LPMS [2:0]		
	Reset value																		0				0	1	0					1	0	0	0
0x04	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.	PVMEN2	PVMEN1	Res.	Res.	PLS[2:0]			PVDE
	Reset value																									0	0			0	0	0	0
0x08	PWR_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EIWUL	UCPD1_DBDIS	UCPD1_STDBY	Res.	Res.	APC	Res.	RRS	Res.	Res.	Res.	EWUP5	EWUP4	EWUP3	EWUP2	EWUP1
	Reset value																	0	0	0			0		0				0	0	0	0	0
0x0C	PWR_CR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBRS	VBE	Res.	Res.	Res.	WP5	WP4	WP3	WP2	WP1
	Reset value																							0	0				0	0	0	0	0
0x10	PWR_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFI	Res.	Res.	Res.	Res.	Res.	Res.	SBF	Res.	Res.	Res.	WUF5	WUF4	WUF3	WUF2	WUF1
	Reset value																	0							0				0	0	0	0	0
0x14	PWR_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVMO2	PVMO1	Res.	Res.	PVDO	VOSF	REGLPF	REGLPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0			0	0	0	0								
0x18	PWR_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBF	Res.	Res.	Res.	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
	Reset value																								0				0	0	0	0	0
0x20	PWR_PUCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU15	Res.	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
0x24	PWR_PDCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD14	Res.	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
0x28	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
0x2C	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	Res.	PD3	PD2	PD1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0
0x30	PWR_PUCRC	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
0x34	PWR_PDCRC	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
0x38	PWR_PUCRD	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 49. PWR 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
0x3C	PWR_PDCRD	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	D1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	PWR_PUCRE	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
0x44	PWR_PDCRE	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
0x48	PWR_PUCRF	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
0x4C	PWR_PDCRF	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
0x50	PWR_PUCRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x54	PWR_PDCRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x80	PWR_CR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R1MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								1								

Refer to *Section 2.2 on page 81* for the register boundary addresses.

7 Reset and clock control (RCC)

7.1 Reset

There are three types of reset, defined as system reset, power reset and RTC domain reset.

7.1.1 Power reset

A power reset is generated when one of the following events occurs:

1. Power-on reset (POR) or Brown-out reset (BOR).
2. when exiting from Standby mode.
3. when exiting from Shutdown mode.

A Brown-out reset, including power-on or power-down reset (POR/PDR), sets all registers to their reset values except the RTC 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 Brown-out reset is generated, resetting all registers except those in the RTC domain.

7.1.2 System reset

A system reset sets all registers to their reset values except the reset flags in the clock control/status register (RCC_CSR) and the registers in the RTC domain.

A system reset is generated when one of the following events occurs:

1. A low level on the NRST pin (external reset)
2. Window watchdog event (WWDG reset)
3. Independent watchdog event (IWDG reset)
4. A software reset (SW reset) (see *Software reset*)
5. Low-power mode security reset (see *Low-power mode security reset*)
6. Option byte loader reset (see *Option byte loader reset*)
7. A Brown-out reset

The reset source can be identified by checking the reset flags in the Control/Status register, RCC_CSR (see *Section 7.4.28: Control/status register (RCC_CSR)*).

NRST pin (external reset)

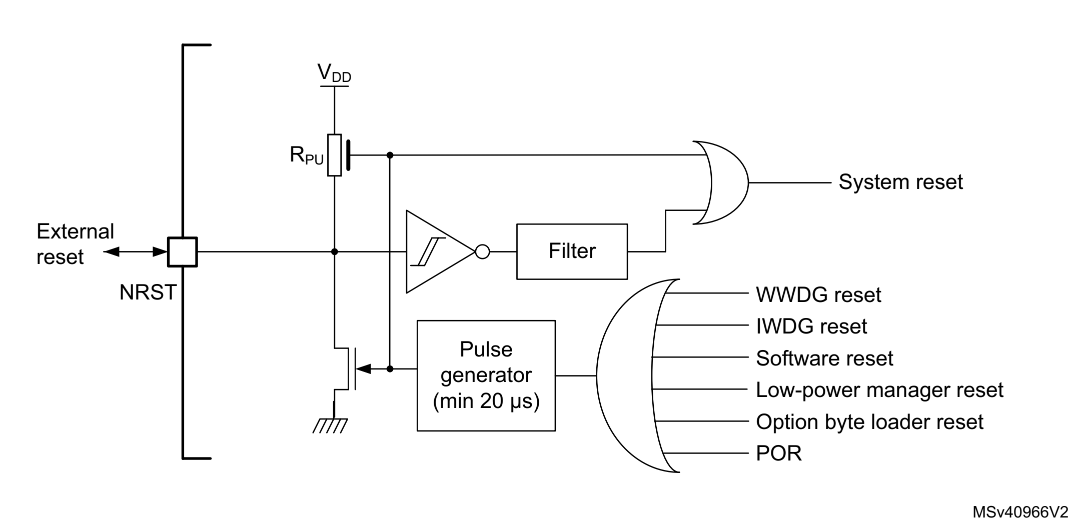
Through specific option bits (NRST_MODE), the NRST pin is configurable for operating as:

- Reset input/output (default at device delivery)
  Any valid reset signal on the pin is propagated to device internal logic and all internal reset sources are externally driven through a pulse generator to this pin. The GPIO functionality (PG10) is not available. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source to be output on the NRST pin. An internal reset holder option can be used, if enabled in the option bytes, to ensure that the pin is pulled low until its voltage meets VIL threshold. This function guarantee the detection of internal reset sources by external components when the line faces a

significant capacitive load. In case on an internal reset, the internal pull-up RPU is deactivated in order to save the power consumption through the pull-up resistor. This mode is always active (independently of the option bytes setting) during each device power-on-reset (until option bytes are loaded): power on the device or wakeup from Shutdown mode.

- Reset input
  In this mode, any valid reset signal on the NRST pin is propagated to device internal logic, but resets generated internally by the device are not visible on the pin. In this configuration, GPIO functionality (PG10) is not available.
- GPIO
  In this mode, the pin can be used as PG10 standard GPIO. The reset function of the pin is not available. Reset is only possible from device internal reset sources and it is not propagated to the pin.

Figure 16. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in Cortex®-M4 with FPU Application Interrupt and Reset Control Register must be set to force a software reset on the device (refer to the STM32F3xx/F4xx/L4xx Cortex®-M4 programming manual (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:

1. 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.
2. 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.
3. 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 (bit 27) is set in the FLASH_CR register. This bit is used to launch the option byte loading by software.

7.1.3 RTC domain reset

The RTC domain has two specific resets.

A RTC domain reset is generated when one of the following events occurs:

1. Software reset, triggered by setting the BDRST bit in the *RTC domain control register (RCC_BDCR)*.
2. V_{DD} or V_{BAT} power on, if both supplies have previously been powered off.

A RTC domain reset only affects the LSE oscillator, the RTC, the Backup registers and the RCC RTC domain control register.

7.2 Clocks

Three different clock sources can be used to drive the system clock (SYSCLK):

- HSI16 (high speed internal)16 MHz RC oscillator clock
- HSE oscillator clock, from 4 to 48 MHz
- PLL clock

The HSI16 is used as system clock source after startup from Reset.

The devices have the following additional clock sources:

- 32 kHz low speed internal RC (LSI RC) which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes.
- 32.768 kHz low speed external crystal (LSE crystal) which optionally drives the real-time clock (RTCCLK).
- RC 48 MHz internal clock sources (HSI48) to potentially drive the USB FS and the RNG.

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 frequency, the APB1 and APB2 domains. The maximum frequency of the AHB, the APB1 and the APB2 domains is 170 MHz.

All the peripheral clocks are derived from their bus clock (HCLK, PCLK1 or PCLK2) except:

- The 48 MHz clock, used for USB device FS, and RNG. This clock is derived (selected by software) from one of the four following sources:
  - PLL “Q” Clock
  - HSI48 internal oscillator

  When available, the HSI48 48 MHz clock can be coupled to the clock recovery system allowing adequate clock connection for the USB OTG FS (Crystal less solution).
- The ADCs clock which is derived (selected by software) from one of the following sources:
  - System clock (SYSCLK)
  - PLL “P” clock
- The U(S)ARTs clocks which are derived (selected by software) from one of the four following sources:
  - System clock (SYSCLK)
  - HSI16 clock
  - LSE clock
  - APB1 or APB2 clock (PCLK1 or PCLK2 depending on which APB is mapped the U(S)ART)

  The wakeup from Stop mode is supported only when the clock is HSI16 or LSE.
- The I^{2}Cs clocks which are derived (selected by software) from one of the three following sources:
  - System clock (SYSCLK)
  - HSI16 clock
  - APB1 clock (PCLK1)

  The wakeup from Stop mode is supported only when the clock is HSI16.
- The SAI1 clock which is derived (selected by software) from one of the following sources:
  - an external clock mapped on I2S_CKIN
  - System clock
  - PLL “Q” clock
  - HSI16 clock
- The QUADSPI kernel clock which is derived (selected by software) from one of the following sources:
  - System clock,
  - PLL “Q” clock
  - HSI16 clock
- The low-power timer (LPTIM1) clock which is derived (selected by software) from one of the five following sources:
  - LSI clock
  - LSE clock
  - HSI16 clock
  - APB1 clock (PCLK1)
  - External clock mapped on LPTIMx_IN1

  The functionality in Stop mode (including wakeup) is supported only when the clock is

LSI or LSE, or in external clock mode.

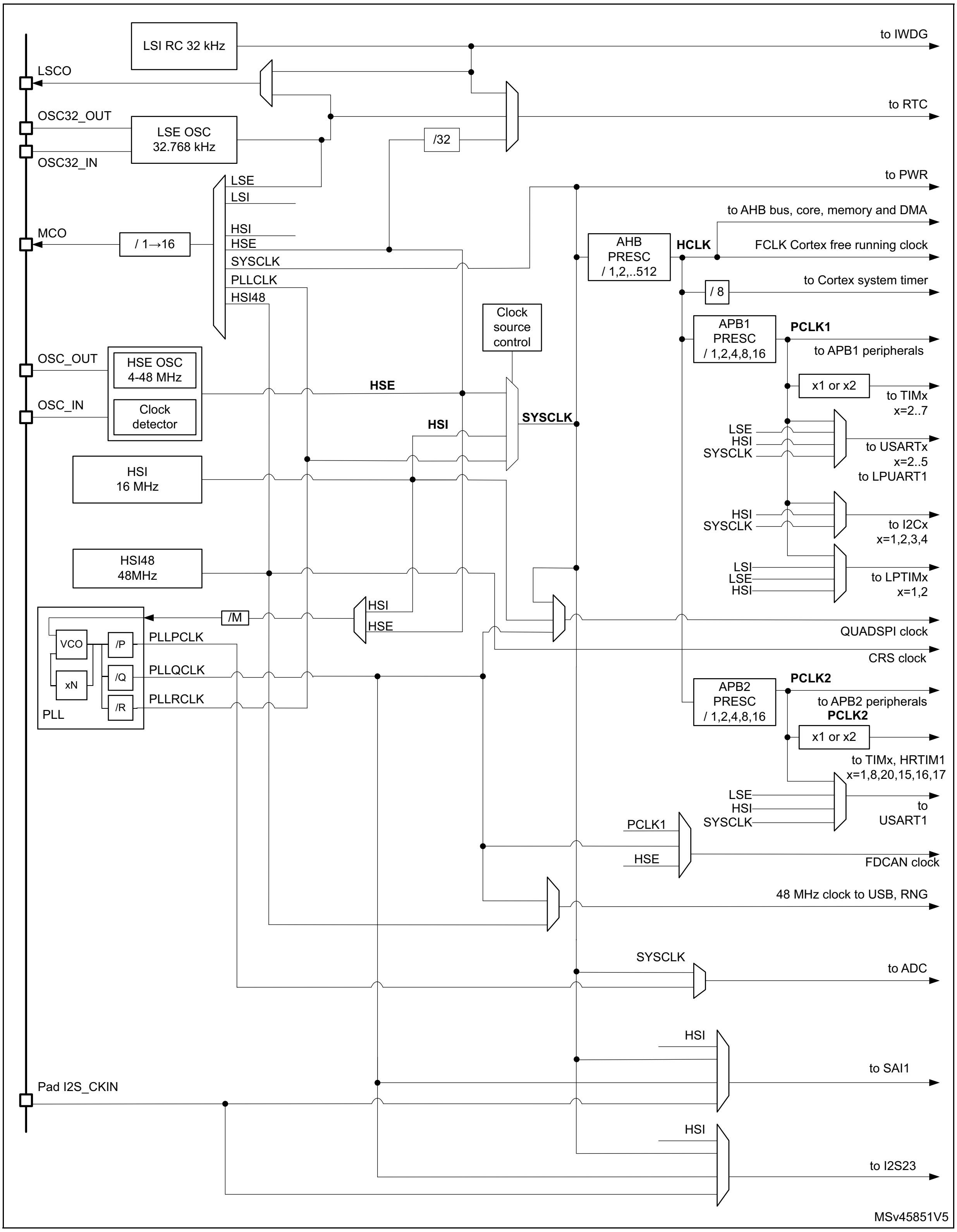
- The RTC clock which is derived (selected by software) from one of the three following sources:
  - LSE clock
  - LSI clock
  - HSE clock divided by 32

  The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE.
- The IWDG clock which is always the LSI clock.
- The UCPD1 clock, which is derived from HSI16 clock.
- The FDCAN1 clock, which is derived (selected by software) from one of the two following sources:
  - HSE clock
  - PLL “Q” clock
  - PCLK clock

The RCC feeds the Cortex® System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or directly with the Cortex® clock (HCLK), configurable in the SysTick Control and Status Register.

FCLK acts as Cortex®-M4 with FPU free-running clock. For more details refer to the Cortex®-M4 programming manual (PM0214).

Figure 17. Clock tree



1. For full details about the internal and external clock source characteristics, please refer to the “Electrical

characteristics" section in your device datasheet.

2. The ADC clock can 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'.

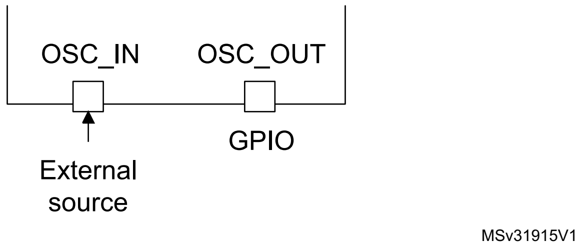
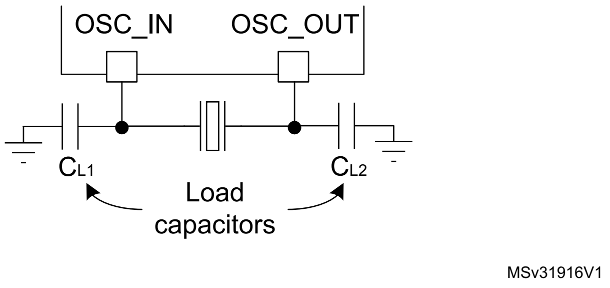
7.2.1 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources:

- HSE external crystal/ceramic resonator
- HSE user external clock

The resonator and the load capacitors have to 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 18. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	 OSC_IN OSC_OUT CL1 CL2 Load capacitors MSv31916V1

External crystal/ceramic resonator (HSE crystal)

The 4 to 48 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in *Figure 18*. Refer to the electrical characteristics section of the *datasheet* for more details.

The HSERDY flag in the *Clock control register (RCC_CR)* indicates if the HSE 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 *Clock interrupt enable register (RCC_CIER)*.

The HSE Crystal can be switched on and off using the HSEON bit in the *Clock control register (RCC_CR)*.

External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 48 MHz. You select this mode by setting the HSEBYP and HSEON bits in the *Clock control register (RCC_CR)*. The external clock signal (square, sinus or triangle) with ~40-60 % duty cycle depending on the frequency (refer to the datasheet) has to drive the OSC_IN pin while the OSC_OUT pin can be used as GPIO. See *Figure 18*.

7.2.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz RC Oscillator.

The HSI16 RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE 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). Refer to *Section 7.3: Low-power modes*. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 7.2.9: Clock security system (CSS)*.

Calibration

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 reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the *Internal clock sources calibration register (RCC_ICSCR)*.

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. You can trim the HSI16 frequency in the application using the HSITRIM[6:0] in the *Internal clock sources calibration register (RCC_ICSCR)*.

For more details on how to measure the HSI16 frequency variation, refer to *Section 7.2.16: Internal/external clock measurement with TIM5/TIM15/TIM16/TIM17*.

The HSIRDY flag in the *Clock control register (RCC_CR)* indicates if the HSI16 oscillator is stable or not. At startup, the HSI16 output clock is not released until this bit is set by hardware.

The HSI16 oscillator can be switched on and off using the HSION bit in the *Clock control register (RCC_CR)*.

The HSI16 signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 7.2.9: Clock security system (CSS) on page 284*.

7.2.3 HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator and can be used directly for USB and for random number generator (RNG).

The internal 48 MHz RC oscillator is mainly dedicated to provide a high precision clock to the USB peripheral by means of a special Clock Recovery System (CRS) circuitry. The CRS can use the LSE or an external signal to automatically and quickly adjust the oscillator frequency on-fly. It is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, the HSI48 RC oscillator runs on its default frequency which is subject to manufacturing process variations.

The HSI48RDY flag in the Clock recovery RC register (RCC_CRRCR) indicates whether the HSI48 RC oscillator is stable or not. At startup, the HSI48 RC oscillator output clock is not released until this bit is set by hardware.

The HSI48 can be switched on and off using the HSI48ON bit in the Clock recovery RC register (RCC_CRRCR).

7.2.4 PLL

The internal PLL can be used to multiply the HSI16 or HSE output clock frequency. The PLL input frequency must be within the range defined in the device datasheet. 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 17: Clock tree*and *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 *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 *PLL configuration register (RCC_PLLCFGR)*.

An interrupt can be generated when the PLL is ready, if enabled in the *Clock interrupt enable register (RCC_CIER)*.

The PLL output frequency must not exceed 170 MHz.

The enable bit of each PLL output clock (PLLPEN, PLLQEN, PLLREN) can be modified at any time without stopping the corresponding PLL. PLLREN cannot be cleared if PLLCLK is used as system clock.

7.2.5 LSE clock

The LSE crystal is a 32.768 kHz Low Speed External crystal or ceramic resonator. It has the advantage of providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in *RTC domain control register (RCC_BDCR)*. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the *RTC 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=00) 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 *RTC 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 *Clock interrupt enable register (RCC_CIER)*.

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. You select this mode by setting the LSEBYP and LSEON bits in the *AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB1SMENR)*. The external clock signal (square, sinus or triangle) with ~50 % duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO. See *Figure 18*.

7.2.6 LSI clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and RTC. The clock frequency is 32 kHz. 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 *Control/status register (RCC_CSR)*.

The LSIRDY flag in the *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 *Clock interrupt enable register (RCC_CIER)*.

7.2.7 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI16 oscillator
- HSE oscillator
- PLL

The system clock maximum frequency is 170 MHz. After a system reset, the HSI16 oscillator 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 which is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *Internal clock sources calibration register (RCC_ICSCR)* indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

To switch from low speed to high speed or from high speed to low speed system clock, it is recommended to use a transition state with medium speed clock, for at least 1 µs.

Clock source switching conditions:

- Switching from HSE or HSI16 to PLL with AHB frequency (HCLK) higher than 80 MHz
- Switching from PLL with HCLK higher than 80 MHz to HSE or HSI16

Transition state:

- Set the AHB prescaler HPRE[3:0] bits to divide the system frequency by 2
- Switch system clock to PLL
- Wait for at least 1 µs and then reconfigure AHB prescaler bits to the needed HCLK frequency

7.2.8 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		
	HSI16	**HSE**	**PLL**
Range 1 Boost mode	16 MHz	48 MHz	170 MHz
Range 1 normal mode	16 MHz	48 MHz	150 MHz
Range 2	16 MHz	26 MHz	26 MHz

7.2.9 Clock security system (CSS)

Clock Security System can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers (TIM1/TIM8/TIM20 and TIM15/16/17) and to the hrtim_sys_flt, and an interrupt is generated to inform the software about the failure (Clock Security System Interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex®-M4 with FPU NMI (Non-Maskable Interrupt) exception vector.

Note: Once the CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and a NMI is automatically generated. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the Clock interrupt clear register (RCC_CICR).

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it 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 HSI16 oscillator, and the disabling of the HSE oscillator. If the HSE clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

7.2.10 Clock security system on LSE

A Clock Security System on LSE can be activated by software writing the LSECSSON bit in the Control/status register (RCC_CSR). 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 enabled) and ready (LSERDY and LSIRDY set by hardware), and after the RTC clock has been selected by RTCSEL.

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 HSI16 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 *Clock interrupt enable register (RCC_CIER)*, *Clock interrupt flag register (RCC_CIFR)*, *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 HSE, with RTCSEL), or take any required action to secure the application.

The frequency of LSE oscillator have to be higher than 30 kHz to avoid false positive CSS detection.

7.2.11 ADC clock

The ADC clock is derived from the system clock, or from the PLL “P” output. It can be divided by the following prescalers values: 1,2,4,6,8,10,12,16,32,64,128 or 256 by configuring the ADCx_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 ADCx_CCR.

If the programmed factor is ‘1’, the AHB prescaler must be set to ‘1’.

Refer to the device datasheet for the maximum ADC clock.

7.2.12 RTC clock

The RTCCLK clock source can be either the HSE/32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the *RTC domain control register (RCC_BDCR)*. This selection cannot be modified without resetting the RTC 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 RTC domain, whereas the HSE and LSI clocks are not. Consequently:

- 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 HSE 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.

7.2.13 Timer clock

The timer clock frequencies are automatically defined by hardware. There are two cases:

1. If the APB prescaler equals 1, the timer clock frequencies are set to the same frequency as that of the APB domain.
2. Otherwise, they are set to twice (×2) the frequency of the APB domain.

7.2.14 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator temporization, the clock is provided to the IWDG.

7.2.15 Clock-out capability

- MCO

  The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. One of eight clock signals can be selected as the MCO clock.

  – LSI
  – LSE
  – SYSCLK
  – HSI16
  – HSI48
  – HSE
  – PLLCLK

  The selection is controlled by the MCOSEL[3:0] bits of the *Clock configuration register (RCC_CFGR)*. The selected clock can be divided with the MCOPRE[2:0] field of the *Clock configuration register (RCC_CFGR)*.

- LSCO

  Another output (LSCO) allows a low speed clock to be output onto the external LSCO pin:

  – LSI
  – LSE

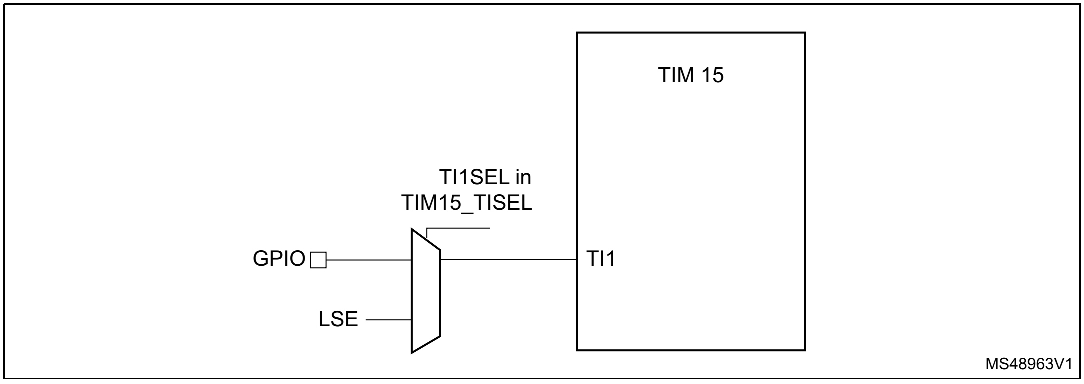
  This output remains available in Stop (Stop 0 and Stop 1) and Standby modes. The selection is controlled by the LSCOSEL, and enabled with the LSCOEN in the *RTC domain control register (RCC_BDCR)*.

The MCO clock output requires the corresponding alternate function selected on the MCO pin, the LSCO pin should be left in default POR state.

7.2.16 Internal/external clock measurement with TIM5/TIM15/TIM16/TIM17

It is possible to indirectly measure the frequency of all on-board clock sources by mean of the TIM5, TIM15, TIM16 or TIM17 channel 1 input capture, as represented on *Figure 19* *Figure 20*, *Figure 21* and *Figure 22*.

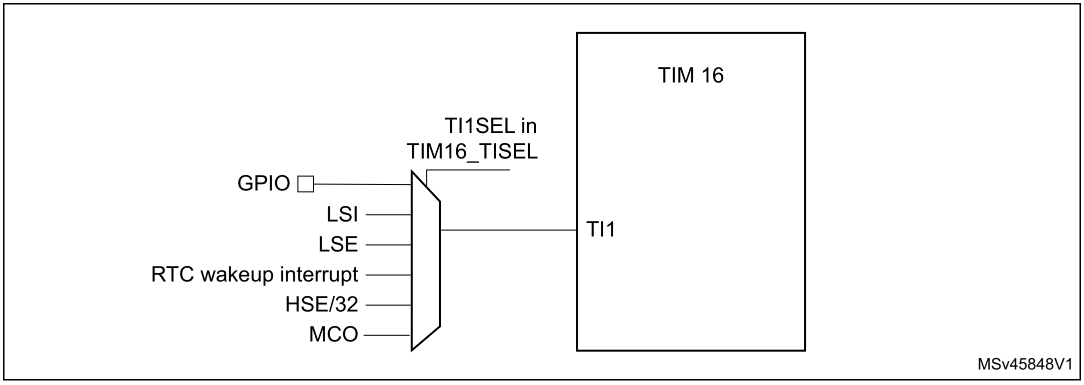
Figure 19. Frequency measurement with TIM15 in capture mode



The input capture channel of the Timer 15 can be a GPIO line or an internal clock of the MCU. The possibilities are the following ones:

- TIM15 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM15 Channel1 is connected to the LSE.

Figure 20. Frequency measurement with TIM16 in capture mode

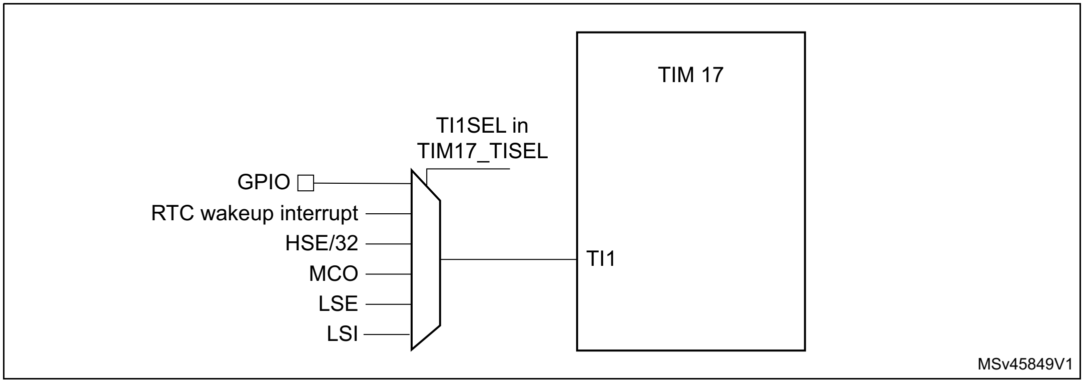


The input capture channel of the Timer 16 can be a GPIO line or an internal clock of the MCU.

The possibilities are the following ones:

- 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 should be enabled.
- TIM16 Channel1 is connected to the HSE/32 clock.
- TIM16 Channel1 is connected to the MCO.

Figure 21. Frequency measurement with TIM17 in capture mode

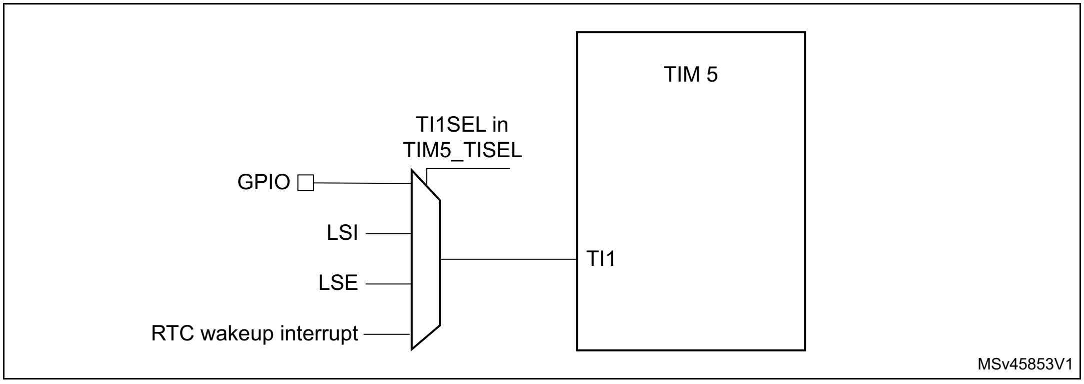


The input capture channel of the Timer 17 can be a GPIO line or an internal clock of the MCU.

The possibilities are the following ones:

- TIM17 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM17 Channel1 is connected to the RTC wakeup interrupt. In this case the RTC interrupt should be enabled.
- TIM17 Channel1 is connected to the HSE/32 Clock.
- TIM17 Channel1 is connected to the microcontroller clock output (MCO), this selection is controlled by the MCOSEL[3:0] bits of the Clock configuration register (RCC_CFGR).
- TIM17 Channel1 is connected to the LSE Clock.
- TIM17 Channel1 is connected to the LSI Clock.

Figure 22. Frequency measurement with TIM5 in capture mode



The input capture channel of the Timer 5 can be a GPIO line or an internal clock of the MCU.

The possibilities are the following ones:

- TIM5 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM5 Channel1 is connected to the LSI Clock.
- TIM5 Channel1 is connected to the LSE Clock.
- TIM5 Channel1 is connected to the RTC wakeup interrupt signal. In this case the RTC interrupt should be enabled.

Calibration of the HSI16

For TIM15 and TIM16, the primary purpose of connecting the LSE to the channel 1 input capture is to be able to precisely measure the HSI16 system clocks (for this, the HSI16 should be used as the system clock source). The number of HSI16 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 ppm's), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing, process, temperature and/or voltage related frequency deviations.

The HSI16 oscillator has dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (e.g. 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, HSE/32 is the better option in order to reach the most precise calibration possible.

Calibration of the LSI

The calibration of the LSI follows the same pattern that for the HSI16, but changing the reference clock. It is necessary to connect LSI clock to the channel 1 input capture of the TIM16. Then define the HSE as system clock source, the number of his 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 (e.g. the HSE/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.

7.2.17 Peripheral clock enable register (RCC_AHBxENR, RCC_APBxENRy)

Each peripheral clock can be enabled by the xxxxEN bit of the RCC_AHBxENR, RCC_APBxENRy registers.

When the peripheral clock is not active, the peripheral registers read or write accesses are not supported.

The enable bit has a synchronization mechanism to create a glitch free clock for the peripheral. After the enable bit is set, there is a 2 clock cycles delay before the clock be active.

Caution: Just after enabling the clock for a peripheral, software must wait for a delay before accessing the peripheral registers.

7.3 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and Low Power Sleep modes stops the CPU clock. The memory interface clocks (Flash and SRAM1, SRAM2 and CCM SRAM interfaces) can be stopped by software

during sleep mode. 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.

- Stop modes (Stop 0 and Stop 1) stops all the clocks in the V_{CORE} domain and disables the PLL, the HSI16, and the HSE oscillators.

  All U(S)ARTs, LPUARTs and I^2Cs have the capability to enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as the clock source for that peripheral).

  All U(S)ARTs and LPUARTs 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 the LSE remains always ON in Stop mode (they do not have the capability to turn on the LSE oscillator).
- Standby and Shutdown modes stops all the clocks in the V_{CORE} domain and disables the PLL, the HSI16, and the HSE oscillators.

The CPU’s deepsleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bits in the DBGMCU_CR register.

When leaving the Stop modes (Stop 0, Stop 1 or standby), the system clock is HSI16.

If a Flash memory programming operation is on going, 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.

7.4 RCC registers

7.4.1 Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 0063

HSEBYP is not affected by reset.

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.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON
						r	rw					rs	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSI RDY	HSI KERON	HSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					r	rw	rw								

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **PLLRDY**: Main PLL clock ready flag

Set by hardware to indicate that the main PLL is locked.

0: PLL unlocked

1: PLL locked

Bit 24 **PLLON**: Main PLL enable

Set and cleared by software to enable the main PLL.

Cleared by hardware when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the PLL clock is used as the system clock.

0: PLL OFF

1: PLL ON

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CSSON**: Clock security system enable

Set by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if a HSE clock failure is detected. This bit is set only and is cleared by reset.

0: Clock security system OFF (clock detector OFF)

1: Clock security system ON (Clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 18 **HSEBYP**: HSE crystal oscillator bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit set, to be used by the device. The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE crystal oscillator not bypassed

1: HSE crystal oscillator bypassed with external clock

Bit 17 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE oscillator not ready

1: HSE oscillator ready

Note: Once the HSEON bit is cleared, HSERDY goes low after 6 HSE clock cycles.

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF

1: HSE oscillator ON

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HSIRDY**: HSI16 clock ready flag

Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

Note: Once the HSION bit is cleared, HSIRDY goes low after 6 HSI16 clock cycles.

Bit 9 **HSIKERON**: HSI16 always enable for peripheral kernels.

Set and cleared by software to force HSI16 ON even in Stop modes. The HSI16 can only feed USARTs and I^{2}Cs peripherals configured with HSI16 as kernel clock. Keeping the HSI16 ON in Stop mode allows to avoid 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 is forced ON even in Stop mode.

Bit 8 **HSION**: HSI16 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSI16 oscillator when entering Stop, Standby or Shutdown mode.

Set by hardware to force the HSI16 oscillator ON when STOPWUCK=1 or HSIASFS = 1 when leaving Stop modes, or in case of failure of the HSE crystal oscillator.

This bit is set by hardware if the HSI16 is used directly or indirectly as system clock.

0: HSI16 oscillator OFF

1: HSI16 oscillator ON

Bits 7:0 Reserved, must be kept at reset value.

7.4.2 Internal clock sources calibration register (RCC_ICSCR)

Address offset: 0x04

Reset value: 0x40XX 00XX

where X 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**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI16.

The default value is 16, which, when added to the HSICAL value, should 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 15:0 Reserved, must be kept at reset value.

7.4.3 Clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0005

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 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 on going.

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.	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.	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	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 is divided by 1
001: MCO is divided by 2
010: MCO is divided by 4
011: MCO is divided by 8
100: MCO is divided by 16
Others: not allowed

Bits 27:24 **MCOSEL[3:0]**: Microcontroller clock output

Set and cleared by software.
0000: MCO output disabled, no clock on MCO
0001: SYSCLK system clock selected
0010: Reserved, must be kept at reset value
0011: HSI16 clock selected
0100: HSE clock selected
0101: Main PLL clock selected
0110: LSI clock selected
0111: LSE clock selected
1000: Internal HSI48 clock selected
Others: Reserved

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:14 Reserved, must be kept at reset value.

Bits 13:11 **PPRE2[2:0]**: APB2 prescaler

Set and cleared by software to control the division factor of the APB2 clock (PCLK2).
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bits 10:8 **PPRE1[2:0]:**APB1 prescaler

Set and cleared by software to control the division factor of the APB1 clock (PCLK1).
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bits 7:4 **HPRE[3:0]**: AHB prescaler

Set and cleared by software to control the division factor of the AHB clock.

Caution: Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details please refer to *Section 6.1.5: Dynamic voltage scaling management*). After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value has been taken into account.

0xxx: SYSCLK not divided
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

Bits 3:2 **SWS[1:0]**: System clock switch status

Set and cleared by hardware to indicate which clock source is used as system clock.
00: Reserved, must be kept at reset value
01: HSI16 oscillator used as system clock
10: HSE used as system clock
11: PLL used as system clock

Bits 1:0 **SW[1:0]**: System clock switch

Set and cleared by software to select system clock source (SYSCLK).
Configured by hardware to force HSI16 oscillator selection when exiting stop and standby modes or in case of failure of the HSE oscillator.
00: Reserved, must be kept at reset value
01: HSI16 selected as system clock
10: HSE selected as system clock
11: PLL selected as system clock

7.4.4 PLL configuration register (RCC_PLLCFGR)

Address offset: 0x0C

Reset value: 0x0000 1000

Access: no wait state, word, half-word and byte access

This register is used to configure the 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
PLLPDIV[4:0]					PLLR[1:0]		PLL REN	Res.	PLLQ[1:0]		PLL QEN	Res.	Res.	PLLP	PLL PEN
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]							PLLM[3:0]				Res.	Res.	PLLSRC[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:27 **PLLPDIV[4:0]**: Main PLLP division factor

Set and cleared by software to control the PLL “P” frequency. PLL “P” output clock frequency = VCO frequency / PLLPDIV.
00000: PLL “P” clock is controlled by the bit PLLP
00001: Reserved.
00010: PLL “P” clock = VCO / 2
....
11111: PLL “P” clock = VCO / 31

Bits 26:25 **PLLR[1:0]**: Main PLL division factor for PLL “R” clock (system clock)

Set and cleared by software to control the frequency of the main PLL output clock PLLCLK. This output can be selected as system clock. These bits can be written only if PLL is disabled.

PLL “R” output clock frequency = VCO frequency / PLLR with PLLR = 2, 4, 6, or 8

00: PLLR = 2

01: PLLR = 4

10: PLLR = 6

11: PLLR = 8

Caution: The software has to set these bits correctly not to exceed 170 MHz on this domain.

Bit 24 **PLLREN**: PLL “R” clock output enable

Set and reset by software to enable the PLL “R” clock output of the PLL (used as system clock).

This bit cannot be written when PLL “R” clock output of the PLL is used as System Clock.

In order to save power, when the PLL “R” clock output of the PLL is not used, the value of PLLREN should be 0.

0: PLL “R” clock output disable

1: PLL “R” clock output enable

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **PLLQ[1:0]**: Main PLL division factor for PLL “Q” clock.

Set and cleared by software to control the frequency of the main PLL output clock PLL “Q” clock. This output can be selected for USB, RNG, SAI (48 MHz clock). These bits can be written only if PLL is disabled.

PLL “Q” output clock frequency = VCO frequency / PLLQ with PLLQ = 2, 4, 6, or 8

00: PLLQ = 2

01: PLLQ = 4

10: PLLQ = 6

11: PLLQ = 8

Caution: The software has to set these bits correctly not to exceed 170 MHz on this domain.

Bit 20 **PLLQEN**: Main PLL “Q” clock output enable

Set and reset by software to enable the PLL “Q” clock output of the PLL.

In order to save power, when the PLL “Q” clock output of the PLL is not used, the value of PLLQEN should be 0.

0: PLL “Q” clock output disable

1: PLL “Q” clock output enable

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **PLLP**: Main PLL division factor for PLL “P” clock.

Set and cleared by software to control the frequency of the main PLL output clock PLL “P” clock. These bits can be written only if PLL is disabled.

When the PLLPDIV[4:0] is set to “00000”PLL “P” output clock frequency = VCO frequency / PLLP with PLLP =7, or 17

0: PLLP = 7

1: PLLP = 17

Caution: The software has to set these bits correctly not to exceed 170 MHz on this domain.

Bit 16 **PLLPEN**: Main PLL PLL **“P”** clock output enable

Set and reset by software to enable the PLL **“P”** clock output of the PLL.
In order to save power, when the PLL **“P”** clock output of the PLL is not used, the value of PLLPEN should be 0.
0: PLL **“P”** clock output disable
1: PLL **“P”** clock output enable

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLN[6:0]**: Main PLL multiplication factor for VCO

Set and cleared by software to control the multiplication factor of the VCO. These bits can be written only when the PLL is disabled.
VCO output frequency = VCO input frequency x PLLN with 8 =< PLLN =< 127
0000000: PLLN = 0 wrong configuration
0000001: PLLN = 1 wrong configuration
...
0000111: PLLN = 7 wrong configuration
0001000: PLLN = 8
0001001: PLLN = 9
...
1111111: PLLN = 127

Caution: The software has to set correctly these bits to assure that the VCO output frequency is within the range defined in the device datasheet.

Bits 7:4 **PLLM[3:0]:** Division factor for the main PLL input clock

Set and cleared by software to divide the PLL input clock before the VCO. These bits can be written only when all PLLs are disabled.
VCO input frequency = PLL input clock frequency / PLLM with 1 <= PLLM <= 16
0000: PLLM = 1
0001: PLLM = 2
0010: PLLM = 3
0011: PLLM = 4
0100: PLLM = 5
0101: PLLM = 6
0110: PLLM = 7
0111: PLLM = 8
1000: PLLSYSM = 9
...
1111: PLLSYSM= 16

Caution: The software has to set these bits correctly to ensure that the VCO input frequency is within the range defined in the device datasheet.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSRC**[1:0]: Main PLL entry clock source

Set and cleared by software to select PLL clock source. These bits can be written only when PLL is disabled.
In order to save power, when no PLL is used, the value of PLLSRC should be 00.
00: No clock sent to PLL
01: No clock sent to PLL
10: HSI16 clock selected as PLL clock entry
11: HSE clock selected as PLL clock entry

7.4.5 Clock interrupt enable register (RCC_CIER)

Address offset: 0x18

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.	HSI48 RDYIE	LSE CSSIE	Res.	Res.	Res.	PLL RDYIE	HSE RDYIE	HSI RDYIE	Res.	LSE RDYIE	LSI RDYIE
					rw	rw				rw	rw	rw		rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSI48RDYIE**: HSI48 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the internal HSI48 oscillator.

0: HSI48 ready interrupt disabled

1: HSI48 ready interrupt enabled

Bit 9 **LSECSSIE**: LSE clock security system interrupt enable

Set and cleared by software to enable/disable interrupt caused by the clock security system 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

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**: HSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization.

0: HSE ready interrupt disabled

1: HSE ready interrupt enabled

Bit 3 **HSIRDYIE**: HSI16 ready interrupt enable

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 Reserved, must be kept at reset value.

Bit 1 **LSERDYIE**: LSE ready interrupt enable

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

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

7.4.6 Clock interrupt flag register (RCC_CIFR)

Address offset: 0x1C

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.	HSI48 RDYF	LSE CSSF	CSSF	Res.	Res.	PLL RDYF	HSE RDYF	HSI RDYF	Res.	LSE RDYF	LSI RDYF
					r	r	r			r	r	r		r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSI48RDYF**: HSI48 ready interrupt flag

Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set in a response to setting the HSI48ON (refer to *Clock recovery RC register (RCC_CRRCR)*). Cleared by software setting the HSI48RDYC bit.

0: No clock ready interrupt caused by the HSI48 oscillator

1: Clock ready interrupt caused by the HSI48 oscillator

Bit 9 **LSECSSF**: LSE Clock security system interrupt flag

Set by hardware when a failure is detected in the LSE oscillator.

Cleared by software setting the LSECSSC bit.

0: No clock security interrupt caused by LSE clock failure

1: Clock security interrupt caused by LSE clock failure

Bit 8 **CSSF**: Clock security system interrupt flag

Set by hardware when a failure is detected in the HSE oscillator.

Cleared by software setting the CSSC bit.

0: No clock security interrupt caused by HSE clock failure

1: Clock security interrupt caused by HSE clock failure

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLLRDYF**: PLL ready interrupt flag

Set by hardware when the PLL locks and PLLRDYDIE is set.
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**: HSE ready interrupt flag

Set by hardware when the HSE clock becomes stable and HSERDYDIE is set.
Cleared by software setting the HSERDYC bit.
0: No clock ready interrupt caused by the HSE oscillator
1: Clock ready interrupt caused by the HSE oscillator

Bit 3 **HSIRDYF**: HSI16 ready interrupt flag

Set by hardware when the HSI16 clock becomes stable and HSIRDYDIE is set in a response to setting the HSION (refer to *Clock control register (RCC_CR)*). 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.
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 Reserved, must be kept at reset value.

Bit 1 **LSERDYF**: LSE ready interrupt flag

Set by hardware when the LSE clock becomes stable and LSERDYDIE is set.
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

Set by hardware when the LSI clock becomes stable and LSIRDYDIE is set.
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

7.4.7 Clock interrupt clear register (RCC_CICR)

Address offset: 0x20

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.	HSI48 RDYC	LSE CSSC	CSSC	Res.	Res.	PLL RDYC	HSE RDYC	HSI RDYC	Res.	LSE RDYC	LSI RDYC
					w	w	w			w	w	w		w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSI48RDYC**: HSI48 oscillator ready interrupt clear

This bit is set by software to clear the HSI48RDYF flag.
0: No effect
1: Clear the HSI48RDYC flag

Bit 9 **LSECSSC**: LSE Clock security system interrupt clear

This bit is set by software to clear the LSECSSF flag.
0: No effect
1: Clear LSECSSF flag

Bit 8 **CSSC**: Clock security system interrupt clear

This bit is set by software to clear the CSSF flag.
0: No effect
1: Clear CSSF flag

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: Clear PLLRDYF flag

Bit 4 **HSERDYC**: HSE ready interrupt clear

This bit is set by software to clear the HSERDYF flag.
0: No effect
1: Clear HSERDYF flag

Bit 3 **HSIRDYC**: HSI16 ready interrupt clear

This bit is set software to clear the HSIRDYF flag.
0: No effect
1: Clear HSIRDYF flag

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYC**: LSE ready interrupt clear

This bit is set by software to clear the LSERDYF flag.
0: No effect
1: LSERDYF cleared

Bit 0 **LSIRDYC**: LSI ready interrupt clear

This bit is set by software to clear the LSIRDYF flag.
0: No effect
1: LSIRDYF cleared

7.4.8 AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x28

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.	FLASH RST	Res.	Res.	Res.	FMAC RST	CORDIC RST	DMAMUX1 RST	DMA2 RST	DMA1 RST
			rw				rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC reset

Set and cleared by software.

0: No effect

1: Reset CRC

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHRST**: Flash memory interface reset

Set and cleared by software. This bit can be activated only when the Flash memory is in power down mode.

0: No effect

1: Reset Flash memory interface

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **FMACRST:** Set and cleared by software

0: No effect

1: Reset FMAC

Bit 3 **CORDICRST:** Set and cleared by software

0: No effect

1: Reset CORDIC

Bit 2 **DMAMUX1RST:** Set and cleared by software.

0: No effect

1: Reset DMAMUX1

Bit 1 **DMA2RST**: DMA2 reset

Set and cleared by software.

0: No effect

1: Reset DMA2

Bit 0 **DMA1RST**: DMA1 reset

Set and cleared by software.

0: No effect

1: Reset DMA1

7.4.9 AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x2C

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.	RNG RST	Res.	AES RST	Res.	Res.	Res.	Res.	DAC4 RST	DAC3 RST	DAC2 RST	DAC1 RST
					rw		rw					rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	ADC345 RST	ADC12 RST	Res.	Res.	Res.	Res.	Res.	Res.	GPIOG RST	GPIOF RST	GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST
	rw	rw							rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RNGRST**: RNG reset
Set and cleared by software.
0: No effect
1: Reset RNG

Bit 25 Reserved, must be kept at reset value.

Bit 24 **AESRST**: AESRST reset
Set and cleared by software.
0: No effect
1: Reset AES

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **DAC4RST**: DAC4 reset
Set and cleared by software.
0: No effect
1: Reset DAC4

Bit 18 **DAC3RST**: DAC3 reset
Set and cleared by software.
0: No effect
1: Reset DAC3

Bit 17 **DAC2RST**: DAC2 reset
Set and cleared by software.
0: No effect
1: Reset DAC2

Bit 16 **DAC1RST**: DAC1 reset
Set and cleared by software.
0: No effect
1: Reset DAC1

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ADC345RST**: ADC345 reset
Set and cleared by software.
0: No effect
1: Reset ADC345

Bit 13 **ADC12RST**: ADC12 reset
Set and cleared by software.
0: No effect
1: Reset ADC12 interface

Bits 12:7 Reserved, must be kept at reset value.

Bit 6 **GPIOGRST**: IO port G reset
Set and cleared by software.
0: No effect
1: Reset IO port G

Bit 5 **GPIOFRST**: IO port F reset
Set and cleared by software.
0: No effect
1: Reset IO port F

Bit 4 **GPIOERST**: IO port E reset
Set and cleared by software.
0: No effect
1: Reset IO port E

Bit 3 **GPIODRST**: IO port D reset
Set and cleared by software.
0: No effect
1: Reset IO port D

Bit 2 **GPIOCRST**: IO port C reset
Set and cleared by software.
0: No effect
1: Reset IO port C

Bit 1 **GPIOBRST**: IO port B reset
Set and cleared by software.
0: No effect
1: Reset IO port B

Bit 0 **GPIOARST**: IO port A reset
Set and cleared by software.
0: No effect
1: Reset IO port A

7.4.10 AHB3 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x30

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.	QSPIRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMC RST
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **QSPIRST**: QUADSPI reset
Set and cleared by software.
0: No effect
1: Reset QUADSPI

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

Bit 0 **FMCRST**: Flexible static memory controller reset
Set and cleared by software.
0: No effect
1: Reset FSMC

7.4.11 APB1 peripheral reset register 1 (RCC_APB1RSTR1)

Address offset: 0x38

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	I2C3 RST	Res.	PWR RST	Res.	Res.	FDCAN RST	Res.	USB RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART3 RST	USART2 RST	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
SPI3 RST	SPI2 RST	Res.	Res.	Res.	Res.	Res.	CRS RST	Res.	Res.	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw						rw			rw	rw	rw	rw	rw	rw

Bit 31 **LPTIM1RST**: Low Power Timer 1 reset

Set and cleared by software.
0: No effect
1: Reset LPTIM1

Bit 30 **I2C3RST**: I2C3 reset

Set and cleared by software.
0: No effect
1: Reset I2C3 interface

Bit 29 Reserved, must be kept at reset value.

Bit 28 **PWRRST**: Power interface reset

Set and cleared by software.
0: No effect
1: Reset PWR

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **FDCANRST**: FDCAN reset

Set and reset by software.
0: No effect
1: Reset the FDCAN

Bit 24 Reserved, must be kept at reset value.

Bit 23 **USBRST**: USB device reset

Set and reset by software.
0: No effect
1: Reset USB device

Bit 22 **I2C2RST**: I2C2 reset

Set and cleared by software.
0: No effect
1: Reset I2C2

Bit 21 **I2C1RST**: I2C1 reset

Set and cleared by software.
0: No effect
1: Reset I2C1

Bit 20 **UART5RST**: UART5 reset

Set and cleared by software.
0: No effect
1: Reset UART5

Bit 19 **UART4RST**: UART4 reset

Set and cleared by software.
0: No effect
1: Reset UART4

Bit 18 **USART3RST**: USART3 reset

Set and cleared by software.
0: No effect
1: Reset USART3

Bit 17 **USART2RST**: USART2 reset
Set and cleared by software.
0: No effect
1: Reset USART2

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3RST**: SPI3 reset
Set and cleared by software.
0: No effect
1: Reset SPI3

Bit 14 **SPI2RST**: SPI2 reset
Set and cleared by software.
0: No effect
1: Reset SPI2

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **CRSRST**: CRS reset
Set and cleared by software.
0: No effect
1: Reset CRS

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **TIM7RST**: TIM7 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM7

Bit 4 **TIM6RST**: TIM6 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM7

Bit 3 **TIM5RST**: TIM5 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM5

Bit 2 **TIM4RST**: TIM3 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM3

Bit 1 **TIM3RST**: TIM3 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM3

Bit 0 **TIM2RST**: TIM2 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM2

7.4.12 APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x3C

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.	UCPD1 RST	Res.	Res.	Res.	Res.	Res.	Res.	I2C4 RST	LP UART1 RST
							rw							rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCPD1RST**: UCPD1 reset

Set and cleared by software.

0: No effect

1: Reset UCPD1

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

Bit 1 **I2C4RST**: I2C4 reset

Set and cleared by software

0: No effect

1: Reset I2C4

Bit 0 **LPUART1RST**: Low-power UART 1 reset

Set and cleared by software.

0: No effect

1: Reset LPUART1

7.4.13 APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x40

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.	HRTIM1 RST	Res.	Res.	Res.	Res.	SAI1 RST	TIM20 RST	Res.	TIM17 RST	TIM16 RST	TIM15RST
					rw					rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI4 RST	USART1 RST	TIM8 RST	SPI1 RST	TIM1 RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYS CFG RST
rw	rw	rw	rw	rw											rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **HRTIM1RST**: HRTIM1 reset
Set and cleared by software.
0: No effect
1: Reset HRTIM1

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **SAI1RST**: Serial audio interface 1 (SAI1) reset
Set and cleared by software.
0: No effect
1: Reset SAI1

Bit 20 **TIM20RST**: TIM20 reset
Set and cleared by software.
0: No effect
1: Reset TIM20

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17RST**: TIM17 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM17 timer

Bit 17 **TIM16RST**: TIM16 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM16 timer

Bit 16 **TIM15RST**: TIM15 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM15 timer

Bit 15 **SPI4RST**: SPI4 reset
Set and cleared by software.
0: No effect
1: Reset SPI4

Bit 14 **USART1RST**: USART1 reset
Set and cleared by software.
0: No effect
1: Reset USART1

Bit 13 **TIM8RST**: TIM8 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM8 timer

Bit 12 **SPI1RST**: SPI1 reset
Set and cleared by software.
0: No effect
1: Reset SPI1

Bit 11 **TIM1RST**: TIM1 timer reset

Set and cleared by software.
0: No effect
1: Reset TIM1 timer

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRST**: SYSCFG + COMP + OPAMP + VREFBUF reset

0: No effect
1: Reset SYSCFG + COMP + OPAMP + VREFBUF

7.4.14 AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x48

Reset value: 0x0000 0100

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.	CRCEN	Res.	Res.	Res.	FLASH EN	Res.	Res.	Res.	FMAC EN	CORDIC EN	DMAM UX1EN	DMA2 EN	DMA1 EN
			rw				rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC clock enable

Set and cleared by software.
0: CRC clock disable
1: CRC clock enable

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHEN**: Flash memory interface clock enable

Set and cleared by software. This bit can be disabled only when the Flash is in power down mode.
0: Flash memory interface clock disable
1: Flash memory interface clock enable

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **FMACEN**: FMAC enable

Set and reset by software.
0: FMAC clock disabled
1: FMAC clock enabled

Bit 3 **CORDICEN**: CORDIC clock enable

Set and reset by software.
0: CORDIC clock disabled
1: CORDIC clock enabled

Bit 2 **DMAMUX1EN**: DMAMUX1 clock enable

Set and reset by software.
0: DMAMUX1 clock disabled
1: DMAMUX1 clock enabled

Bit 1 **DMA2EN**: DMA2 clock enable

Set and cleared by software.
0: DMA2 clock disable
1: DMA2 clock enable

Bit 0 **DMA1EN**: DMA1 clock enable

Set and cleared by software.
0: DMA1 clock disable
1: DMA1 clock enable

7.4.15 AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x4C

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.	RNG EN	Res.	AES EN	Res.	Res.	Res.	Res.	DAC4 EN	DAC3 EN	DAC2 EN	DAC1 EN
					rw		rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADC345 EN	ADC12 EN	Res.	Res.	Res.	Res.	Res.	Res.	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
	rw	rw							rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RNGEN**: RNG enable

Set and cleared by software.
0: RNG disabled
1: RNG enabled

Bit 25 Reserved, must be kept at reset value.

Bit 24 **AESEN**: AES clock enable

Set and cleared by software.
0: AES clock disabled
1: AES clock enabled

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **DAC4EN:** DAC4 clock enable

Set and cleared by software.
0: DAC4 clock disabled
1: DAC4 clock enabled

Bit 18 **DAC3EN:** DAC3 clock enable

Set and cleared by software.
0: DAC3 clock disabled
1: DAC3 clock enabled

Bit 17 **DAC2EN:** DAC2 clock enable

Set and cleared by software.
0: DAC2 clock disabled
1: DAC2 clock enabled

Bit 16 **DAC1EN:** DAC1 clock enable

Set and cleared by software.
0: DAC1 clock disabled
1: DAC1 clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ADC345EN**: ADC345 clock enable

Set and cleared by software

0: ADC345 clock disabled
1: ADC345 clock enabled

Bit 13 **ADC12EN:** ADC12 clock enable

Set and cleared by software.
0: ADC12 clock disabled
1: ADC12 clock enabled

Bits 12:7 Reserved, must be kept at reset value.

Bit 6 **GPIOGEN:** IO port G clock enable

Set and cleared by software.
0: IO port G clock disabled
1: IO port G clock enabled

Bit 5 **GPIOFEN:** IO port F clock enable

Set and cleared by software.
0: IO port F clock disabled
1: IO port F clock enabled

Bit 4 **GPIOEEN:** IO port E clock enable

Set and cleared by software.
0: IO port E clock disabled
1: IO port E clock enabled

Bit 3 **GPIODEN:** IO port D clock enable

Set and cleared by software.
0: IO port D clock disabled
1: IO port D clock enabled

Bit 2 **GPIOCEN:** IO port C clock enable

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

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

Set and cleared by software.
0: IO port A clock disabled
1: IO port A clock enabled

7.4.16 AHB3 peripheral clock enable register(RCC_AHB3ENR)

Address offset: 0x50

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.	QSPIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMC EN
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **QSPIEN**: QUADSPI memory interface clock enable

Set and cleared by software.
0: QUADSPI clock disable
1: QUADSPI clock enable

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

Bit 0 **FMCEN**: Flexible static memory controller clock enable

Set and cleared by software.
0: FSMC clock disable
1: FSMC clock enable

7.4.17 APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address: 0x58

Reset value: 0x0000 0400

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	I2C3 EN	Res.	PWR EN	Res.	Res.	FDCAN EN	Res.	USB EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	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
SPI3 EN	SPI2 EN	Res.	Res.	WWDG EN	RTCAPB EN	Res.	CRS EN	Res.	Res.	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rs	rw		rw			rw	rw	rw	rw	rw	rw

Bit 31 **LPTIM1EN**: Low power timer 1 clock enable

Set and cleared by software.
0: LPTIM1 clock disabled
1: LPTIM1 clock enabled

Bit 30 **I2C3EN**: I2C3 clock enable

Set and cleared by software.
0: I2C3 clock disabled
1: I2C3 clock enabled

Bit 29 Reserved, must be kept at reset value.

Bit 28 **PWREN**: Power interface clock enable

Set and cleared by software.
0: Power interface clock disabled
1: Power interface clock enabled

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **FDCANEN:** FDCAN clock enable

Set and cleared by software.
0: FDCAN clock disabled
1: FDCAN clock enabled

Bit 24 Reserved, must be kept at reset value.

Bit 23 **USBEN**: USB device clock enable

Set and cleared by software.
0: USB device clock disabled
1: USB device clock enabled

Bit 22 **I2C2EN**: I2C2 clock enable

Set and cleared by software.
0: I2C2 clock disabled
1: I2C2 clock enabled

Bit 21 **I2C1EN**: I2C1 clock enable
Set and cleared by software.
0: I2C1 clock disabled
1: I2C1 clock enabled

Bit 20 **UART5EN**: UART5 clock enable
Set and cleared by software.
0: UART5 clock disabled
1: UART5 clock enabled

Bit 19 **UART4EN**: UART4 clock enable
Set and cleared by software.
0: UART4 clock disabled
1: UART4 clock enabled

Bit 18 **USART3EN**: USART3 clock enable
Set and cleared by software.
0: USART3 clock disabled
1: USART3 clock enabled

Bit 17 **USART2EN**: USART2 clock enable
Set and cleared by software.
0: USART2 clock disabled
1: USART2 clock enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3EN**: SPI3 clock enable
Set and cleared by software.
0: SPI3 clock disabled
1: SPI3 clock enabled

Bit 14 **SPI2EN**: SPI2 clock enable
Set and cleared by software.
0: SPI2 clock disabled
1: SPI2 clock enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: Window watchdog clock enable
Set by software to enable the window watchdog clock. Reset by hardware system reset. This bit can also be set by hardware if the WWDG_SW option bit is reset.
0: Window watchdog clock disabled
1: Window watchdog clock enabled

Bit 10 **RTCAPBEN**: RTC APB clock enable
Set and cleared by software
0: RTC APB clock disabled
1: RTC APB clock enabled

Bit 9 Reserved, must be kept at reset value.

Bit 8 **CRSEN**: CRS Recovery System clock enable
Set and cleared by software.
0: CRS clock disabled
1: CRS clock enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **TIM7EN**: TIM7 timer clock enable

Set and cleared by software.
0: TIM7 clock disabled
1: TIM7 clock enabled

Bit 4 **TIM6EN**: TIM6 timer clock enable

Set and cleared by software.
0: TIM6 clock disabled
1: TIM6 clock enabled

Bit 3 **TIM5EN**: TIM5 timer clock enable

Set and cleared by software.
0: TIM5 clock disabled
1: TIM5 clock enabled

Bit 2 **TIM4EN**: TIM4 timer clock enable

Set and cleared by software.
0: TIM4 clock disabled
1: TIM4 clock enabled

Bit 1 **TIM3EN**: TIM3 timer clock enable

Set and cleared by software.
0: TIM3 clock disabled
1: TIM3 clock enabled

Bit 0 **TIM2EN**: TIM2 timer clock enable

Set and cleared by software.
0: TIM2 clock disabled
1: TIM2 clock enabled

7.4.18 APB1 peripheral clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x5C

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.	UCPD1 EN	Res.	Res.	Res.	Res.	Res.	Res.	I2C4EN	LP UART1 EN
							rw							rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCPD1EN:** UCPD1 clock enable
Set and cleared by software.
0: UCPD1 clock disable
1: UCPD1 clock enable

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

Bit 1 I**2C4EN**: I2C4 clock enable
Set and cleared by software
0: I2C4 clock disabled
1: I2C4 clock enabled

Bit 0 **LPUART1EN**: Low power UART 1 clock enable
Set and cleared by software.
0: LPUART1 clock disable
1: LPUART1 clock enable

7.4.19 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x60

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.	HRTIM1 EN	Res.	Res.	Res.	Res.	SAI1 EN	TIM20 EN	Res.	TIM 17EN	TIM16 EN	TIM15 EN
					rw					rw	rw		rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPI4 EN	USART1 EN	TIM8 EN	SPI1 EN	TIM1 EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYS CFGEN
rw	rw	rw	rw	rw											rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **HRTIM1EN**: HRTIM1 clock enable

Set and cleared by software.
0: HRTIM1 clock disabled
1: HRTIM1 clock enable

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **SAI1EN**: SAI1 clock enable

Set and cleared by software.
0: SAI1 clock disabled
1: SAI1 clock enabled

Bit 20 **TIM20EN**: TIM20 timer clock enable

Set and cleared by software.
0: TIM20 clock disabled
1: TIM20 clock enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: TIM17 timer clock enable

Set and cleared by software.
0: TIM17 timer clock disabled
1: TIM17 timer clock enabled

Bit 17 **TIM16EN**: TIM16 timer clock enable

Set and cleared by software.
0: TIM16 timer clock disabled
1: TIM16 timer clock enabled

Bit 16 **TIM15EN**: TIM15 timer clock enable

Set and cleared by software.
0: TIM15 timer clock disabled
1: TIM15 timer clock enabled

Bit 15 **SPI4EN**: SPI4 clock enable
Set and cleared by software.
0: SPI4 clock disabled
1: SPI4 clock enabled

Bit 14 **USART1EN**: USART1clock enable
Set and cleared by software.
0: USART1clock disabled
1: USART1clock enabled

Bit 13 **TIM8EN**: TIM8 timer clock enable
Set and cleared by software.
0: TIM8 timer clock disabled
1: TIM8 timer clock enabled

Bit 12 **SPI1EN**: SPI1 clock enable
Set and cleared by software.
0: SPI1 clock disabled
1: SPI1 clock enabled

Bit 11 **TIM1EN**: TIM1 timer clock enable
Set and cleared by software.
0: TIM1 timer clock disabled
1: TIM1P timer clock enabled

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGEN**: SYSCFG + COMP + VREFBUF + OPAMP clock enable
Set and cleared by software.
0: SYSCFG + COMP + VREFBUF + OPAMP clock disabled
1: SYSCFG + COMP + VREFBUF + OPAMP clock enabled

7.4.20 AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB1SMENR)

Address offset: 0x68

Reset value: 0x0000 130F

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.	CRCSM EN	Res.	Res.	SRAM1 SMEN	FLASH SMEN	Res.	Res.	Res.	FMACSM EN	CORDICSM EN	DMAMUX1 SMEN	DMA2 SMEN	DMA1 SMEN
			rw			rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CRCSMEN**: CRC clocks enable during Sleep and Stop modes

Set and cleared by software.

0: CRC clocks disabled by the clock gating[1] during Sleep and Stop modes

1: CRC clocks enabled by the clock gating[1] during Sleep and Stop modes

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAM1SMEN**: SRAM1 interface clocks enable during Sleep and Stop modes

Set and cleared by software.

0: SRAM1 interface clocks disabled by the clock gating[1] during Sleep and Stop modes

1: SRAM1 interface clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 8 **FLASHSMEN**: Flash memory interface clocks enable during Sleep and Stop modes

Set and cleared by software.

0: Flash memory interface clocks disabled by the clock gating[1] during Sleep and Stop modes

1: Flash memory interface clocks enabled by the clock gating[1] during Sleep and Stop modes

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **FMACSMEN**: FMACSM clock enable.

Set and cleared by software.

0: FMACSM clocks disabled by the clock gating[1] during Sleep and Stop modes

1: FMACSM clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 3 **CORDICSMEN**: CORDICSM clock enable.

Set and cleared by software.

0: CORDICSM clocks disabled.

1: CORDICSM clocks enabled.

Bit 2 **DMAMUX1SMEN**: DMAMUX1 clock enable during Sleep and Stop modes.

Set and cleared by software.

0: DMAMUX1 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: DMAMUX1 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 1 **DMA2SMEN**: DMA2 clocks enable during Sleep and Stop modes

Set and cleared by software during Sleep mode.

0: DMA2 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: DMA2 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 0 **DMA1SMEN**: DMA1 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: DMA1 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: DMA1 clocks enabled by the clock gating[1] during Sleep and Stop modes

[1] This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

7.4.21 AHB2 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB2SMENR)

Address offset: 0x6C

Reset value: 0x050F 667F

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.	RNG EN	Res.	AESM EN	Res.	Res.	Res.	Res.	DAC4 SMEN	DAC3 SMEN	DAC2 SMEN	DAC1 SMEN
					rw		rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADC345 SMEN	ADC12 SMEN	Res.	Res.	SRAM2 SMEN	CCMSRAM SMEN	Res.	Res.	GPIOG SMEN	GPIOF SMEN	GPIOE SMEN	GPIOD SMEN	GPIOC SMEN	GPIOB SMEN	GPIOA SMEN
	rw	rw			rw	rw			rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RNGEN**: RNG enable

Set and cleared by software.
0: RNG disabled
1: RNG enabled

Bit 25 Reserved, must be kept at reset value.

Bit 24 **AESMEN**: AESM clocks enable

Set and cleared by software.
0: AESM clocks disabled
1: AESM clocks enabled

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **DAC4SMEN:** DAC4 clock enable

Set and cleared by software.
0: DAC4 clock disabled
1: DAC4 clock enabled during sleep and stop modes

Bit 18 **DAC3SMEN:** DAC3 clock enable

Set and cleared by software.
0: DAC3 clock disabled
1: DAC3 clock enabled during sleep and stop modes

Bit 17 **DAC2SMEN:** DAC2 clock enable

Set and cleared by software.
0: DAC2 clock disabled
1: DAC2 clock enabled during sleep and stop modes

Bit 16 **DAC1SMEN:** DAC1 clock enable

Set and cleared by software.
0: DAC1 clock disabled
1: DAC1 clock enabled during sleep and stop modes

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ADC345SMEN:** ADC345 clock enable
Set and cleared by software.
0: ADC345 clock disabled
1: ADC345 clock enabled

Bit 13 **ADC12SMEN:** ADC12 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: ADC12 clocks disabled by the clock gating[1] during Sleep and Stop modes
1: ADC12 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SRAM2SMEN**: SRAM2 interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SRAM2 interface clocks disabled by the clock gating[1] during Sleep and Stop modes
1: SRAM2 interface clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 9 **CCMSRAMSMEN:** CCM SRAM interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: CCM SRAM interface clocks disabled by the clock gating[1] during Sleep and Stop modes
1: CCM SRAM interface clocks enabled by the clock gating[1] during Sleep and Stop modes

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **GPIOGSMEN:** IO port G clocks enable during Sleep and Stop modes
Set and cleared by software.
0: IO port G clocks disabled by the clock gating[1] during Sleep and Stop modes
1: IO port G clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 5 **GPIOFSMEN:** IO port F clocks enable during Sleep and Stop modes
Set and cleared by software.
0: IO port F clocks disabled by the clock gating[1] during Sleep and Stop modes
1: IO port F clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 4 **GPIOESMEN:** IO port E clocks enable during Sleep and Stop modes
Set and cleared by software.
0: IO port E clocks disabled by the clock gating[1] during Sleep and Stop modes
1: IO port E clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 3 **GPIODSMEN:** IO port D clocks enable during Sleep and Stop modes
Set and cleared by software.
0: IO port D clocks disabled by the clock gating[1] during Sleep and Stop modes
1: IO port D clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 2 **GPIOCSMEN:** IO port C clocks enable during Sleep and Stop modes
Set and cleared by software.
0: IO port C clocks disabled by the clock gating[1] during Sleep and Stop modes
1: IO port C clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 1 **GPIOBSMEN:** IO port B clocks enable during Sleep and Stop modes
Set and cleared by software.
0: IO port B clocks disabled by the clock gating[1] during Sleep and Stop modes
1: IO port B clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 0 **GPIOASMEN:** IO port A clocks enable during Sleep and Stop modes
Set and cleared by software.
0: IO port A clocks disabled by the clock gating[1] during Sleep and Stop modes
1: IO port A clocks enabled by the clock gating[1] during Sleep and Stop modes

[1] This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

7.4.22 AHB3 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR)

Address offset: 0x70

Reset value: 0x0000 0101

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.	QSPISMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMC SMEN
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **QSPISMEN**: QUADSPI memory interface clock enable during Sleep and Stop modes

Set and cleared by software.

0: QUADSPI clock disabled by the clock gating[1] during Sleep and Stop modes

1: QUADSPI clock enabled by the clock gating[1] during Sleep and Stop modes

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

Bit 0 **FMCSMEN**: Flexible static memory controller clocks enable during Sleep and Stop modes

Set and cleared by software.

0: FSMC clocks disabled by the clock gating[1] during Sleep and Stop modes

1: FSMC clocks enabled by the clock gating[1] during Sleep and Stop modes

[1] This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode

7.4.23 APB1 peripheral clocks enable in Sleep and Stop modes register 1 (RCC_APB1SMENR1)

Address: 0x78

Reset value: 0xD2FE CD3F

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	I2C3 SMEN	Res.	PWR SMEN	Res.	Res.	FDCAN SMEN	Res.	USB SMEN	I2C2 SMEN	I2C1 SMEN	UART5 SMEN	UART4 SMEN	USART 3SMEN	USART 2SMEN	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
SPI3 SMEN	SPI2 SMEN	Res.	Res.	WWDG SMEN	RTCAPB SMEN	Res.	CRS SMEN	Res.	Res.	TIM7 SMEN	TIM6 SMEN	TIM5 SMEN	TIM4 SMEN	TIM3 SMEN	TIM2 SMEN
rw	rw			rw	rw		rw			rw	rw	rw	rw	rw	rw

Bit 31 **LPTIM1SMEN**: Low power timer 1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: LPTIM1 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: LPTIM1 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 30 **I2C3SMEN**: I2C3 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: I2C3 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: I2C3 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 29 Reserved, must be kept at reset value.

Bit 28 **PWRSMEN**: Power interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: Power interface clocks disabled by the clock gating(1) during Sleep and Stop modes
1: Power interface clocks enabled by the clock gating(1) during Sleep and Stop modes

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **FDCANSMEN:** FDCAN clocks enable during Sleep and Stop modes
Set and cleared by software.
0: FDCAN clocks disabled by the clock gating(1) during Sleep and Stop modes
1: FDCAN clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 24 Reserved, must be kept at reset value.

Bit 23 **USBSMEN**: USB device clocks enable during Sleep and Stop modes
Set and cleared by software.
0: USB device clocks disabled by the clock gating(1) during Sleep and Stop modes
1: USB device clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 22 **I2C2SMEN**: I2C2 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: I2C2 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: I2C2 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 21 **I2C1SMEN**: I2C1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: I2C1 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: I2C1 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 20 **UART5SMEN**: UART5 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: UART5 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: UART5 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 19 **UART4SMEN**: UART4 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: UART4 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: UART4 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 18 **USART3SMEN**: USART3 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: USART3 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: USART3 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 17 **USART2SMEN**: USART2 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: USART2 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: USART2 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3SMEN**: SPI3 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: SPI3 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: SPI3 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 14 **SPI2SMEN**: SPI2 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: SPI2 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: SPI2 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGSMEN**: Window watchdog clocks enable during Sleep and Stop modes

Set and cleared by software. This bit is forced to ‘1’ by hardware when the hardware WWDG option is activated.

0: Window watchdog clocks disabled by the clock gating[1] during Sleep and Stop modes

1: Window watchdog clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 10 **RTCAPBSMEN**: RTC APB clock enable during Sleep and Stop modes

Set and cleared by software

0: RTC APB clock disabled by the clock gating[1] during Sleep and Stop modes

1: RTC APB clock enabled by the clock gating[1] during Sleep and Stop modes

Bit 9 Reserved, must be kept at reset value.

Bit 8 **CRSSMEN**: CRS timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: CRS clocks disabled by the clock gating[1] during Sleep and Stop modes

1: CRS clocks enabled by the clock gating[1] during Sleep and Stop modes

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **TIM7SMEN**: TIM7 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM7 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: TIM7 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 4 **TIM6SMEN**: TIM6 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM6 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: TIM6 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 3 **TIM5SMEN**: TIM5 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM5 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: TIM5 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 2 **TIM4SMEN**: TIM4 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM4 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: TIM4 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 1 **TIM3SMEN**: TIM3 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM3 clocks disabled by the clock gating[1] during Sleep and Stop modes
1: TIM3 clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 0 **TIM2SMEN**: TIM2 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM2 clocks disabled by the clock gating[1] during Sleep and Stop modes
1: TIM2 clocks enabled by the clock gating[1] during Sleep and Stop modes

[1] This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

7.4.24 APB1 peripheral clocks enable in Sleep and Stop modes register 2 (RCC_APB1SMENR2)

Address offset: 0x7C

Reset value: 0x0000 0103

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.	UCPD1 SMEN	Res.	Res.	Res.	Res.	Res.	Res.	I2C4 SMEN	LP UART1 SMEN
							rw							rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCPD1SMEN**: UCPD1 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: UCPD1 clocks disabled by the clock gating[1] during Sleep and Stop modes
1: UCPD1 clocks enabled by the clock gating[1] during Sleep and Stop modes

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

Bit 1 **I2C4SMEN**: I2C4 clocks enable during Sleep and Stop modes

Set and cleared by software

0: I2C4 clocks disabled by the clock gating[2] during Sleep and Stop modes
1: I2C4 clock enabled by the clock gating[1] during Sleep and Stop modes

Bit 0 **LPUART1SMEN**: Low power UART 1 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: LPUART1 clocks disabled by the clock gating[1] during Sleep and Stop modes
1: LPUART1 clocks enabled by the clock gating[1] during Sleep and Stop modes

[1] This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

[2] This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

7.4.25 APB2 peripheral clocks enable in Sleep and Stop modes register (RCC_APB2SMENR)

Address: 0x80

Reset value: 0x0437 F801

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.	HRTIM1 SMEN	Res.	Res.	Res.	Res.	SAI1 SMEN	TIM20 SMEN	Res.	TIM17 SMEN	TIM16 SMEN	TIM15 SMEN
					rw					rw	rw		rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPI4 SMEN	USART1 SMEN	TIM8 SMEN	SPI1 SMEN	TIM1 SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYS CFG SMEN
rw	rw	rw	rw	rw											rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **HRTIM1SMEN**: HRTIM1 timer clocks enable during Sleep and Stop modes
Set and cleared by software.
0: HRTIM1 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: HRTIM1 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **SAI1SMEN**: SAI1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SAI1 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: SAI1 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 20 **TIM20SMEN**: TIM20 timer clocks enable during Sleep and Stop modes
Set and cleared by software.
0: TIM20 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: TIM20 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN**: TIM17 timer clocks enable during Sleep and Stop modes
Set and cleared by software.
0: TIM17 timer clocks disabled by the clock gating(1) during Sleep and Stop modes
1: TIM17 timer clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 17 **TIM16SMEN**: TIM16 timer clocks enable during Sleep and Stop modes
Set and cleared by software.
0: TIM16 timer clocks disabled by the clock gating(1) during Sleep and Stop modes
1: TIM16 timer clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 16 **TIM15SMEN**: TIM15 timer clocks enable during Sleep and Stop modes
Set and cleared by software.
0: TIM15 timer clocks disabled by the clock gating(1) during Sleep and Stop modes
1: TIM15 timer clocks enabled by the clock gating(1) during Sleep and Stop mode

Bit 15 **SPI4SMEN**: SPI4 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: SPI4 clocks disabled by the clock gating[1] during Sleep and Stop modes

1: SPI4 clocks enabled by the clock gating[1] during Sleep and Stop mode

Bit 14 **USART1SMEN**: USART1clocks enable during Sleep and Stop modes

Set and cleared by software.

0: USART1clocks disabled by the clock gating[1] during Sleep and Stop modes

1: USART1clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 13 T**IM8SMEN**: TIM8 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM8 timer clocks disabled by the clock gating[1] during Sleep and Stop modes

1: TIM8 timer clocks enabled by the clock gating[1] during Sleep and Stop modes

Bit 12 **SPI1SMEN**: SPI1 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: SPI1 clocks disabled by the clock gating during[1] Sleep and Stop modes

1: SPI1 clocks enabled by the clock gating during[1] Sleep and Stop modes

Bit 11 **TIM1SMEN**: TIM1 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM1 timer clocks disabled by the clock gating[1] during Sleep and Stop modes

1: TIM1P timer clocks enabled by the clock gating[1] during Sleep and Stop modes

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGSMEN**: SYSCFG + COMP + VREFBUF + OPAMP clocks enable during Sleep and Stop modes

Set and cleared by software.

0: SYSCFG + COMP + VREFBUF + OPAMP clocks disabled by the clock gating[1] during Sleep and Stop modes

1: SYSCFG + COMP + VREFBUF + OPAMP clocks enabled by the clock gating[1] during Sleep and Stop modes

[1] This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

7.4.26 Peripherals independent clock configuration register (RCC_CCIPR)

Address: 0x88

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
ADC345SEL[1:0]		ADC12SEL[1:0]		CLK48SEL[1:0]		FDCANSEL[1:0]		I2S23SEL[1:0]		SAI1SEL[1:0]		LPTIM1SEL[1:0		I2C3SEL[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
I2C2SEL[1:0]		I2C1SEL[1:0]		LPUART1SEL [1:0]		UART5SEL [1:0]		UART4SEL [1:0]		USART3SEL [1:0]		USART2SEL [1:0]		USART1SEL [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **ADC345SEL[1:0]**: ADC3/4/5 clock source selection

These bits are set and cleared by software to select the clock source used by the ADC345 interface.

00: No clock selected
01: PLL “P” clock selected as ADC345 clock
10: System clock selected as ADC3/4/5 clock
11: Reserved.

Bits 29:28 **ADC12SEL[1:0]**: ADC1/2 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: PLL “P” clock selected as ADC1/2 clock
10: System clock selected as ADC1/2 clock
11: Reserved

Bits 27:26 **CLK48SEL[1:0]**: 48 MHz clock source selection

These bits are set and cleared by software to select the 48 MHz clock source used by USB device FS and RNG.

00: HSI48 clock selected as 48 MHz clock
01: Reserved
10: PLL “Q” clock (PLL48M1CLK) selected as 48 MHz clock
11: Reserved, must be kept at reset value

Bits 25:24 **FDCANSEL[1:0]** clock source selection

These bits are set and cleared by software to select the FDCAN clock source.

00: HSE clock selected as FDCAN clock
01: PLL “Q” clock selected as FDCAN clock
10: PCLK clock selected as FDCAN clock
11: Reserved, must be kept at reset value.

Bits 23:22 **I2S23SEL[1:0]:** clock source selection

These bits are set and cleared by software to select the I2S23 clock source.

00: System clock selected as I2S23 clock
01: PLL “Q” clock selected as I2S23 clock
10: Clock provided on I2S_CKIN pin is selected as I2S23 clock
11: HSI16 clock selected as I2S23 clock.

Bits 21:20 **SAI1SEL[1:0]:** clock source selection

These bits are set and cleared by software to select the SAI clock source.

00: System clock selected as SAI clock
01: PLL “Q” clock selected as SAI clock
10: Clock provided on I2S_CKIN pin selected as SAI clock
11: HSI16 clock selected as SAI clock

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 as LPTIM1 clock
01: LSI clock selected as LPTIM1 clock
10: HSI16 clock selected as LPTIM1 clock
11: LSE clock selected as LPTIM1 clock

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 as I2C3 clock
01: System clock (SYSCLK) selected as I2C3 clock
10: HSI16 clock selected as I2C3 clock
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 as I2C2 clock
01: System clock (SYSCLK) selected as I2C2 clock
10: HSI16 clock selected as I2C2 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 as I2C1 clock
01: System clock (SYSCLK) selected as I2C1 clock
10: HSI16 clock selected as I2C1 clock
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 as LPUART1 clock
01: System clock (SYSCLK) selected as LPUART1 clock
10: HSI16 clock selected as LPUART1 clock
11: LSE clock selected as LPUART1 clock

Bits 9:8 **UART5SEL[1:0]**: UART5 clock source selection

These bits are set and cleared by software to select the UART5 clock source.
00: PCLK selected as UART5 clock
01: System clock (SYSCLK) selected as UART5 clock
10: HSI16 clock selected as UART5 clock
11: LSE clock selected as UART5 clock

Bits 7:6 **UART4SEL[1:0]**: UART4 clock source selection

This bit is set and cleared by software to select the UART4 clock source.
00: PCLK selected as UART4 clock
01: System clock (SYSCLK) selected as UART4 clock
10: HSI16 clock selected as UART4 clock
11: LSE clock selected as UART4 clock

Bits 5:4 **USART3SEL[1:0]**: USART3 clock source selection

This bit is set and cleared by software to select the USART3 clock source.
00: PCLK selected as USART3 clock
01: System clock (SYSCLK) selected as USART3 clock
10: HSI16 clock selected as USART3 clock
11: LSE clock selected as USART3 clock

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 as USART2 clock
01: System clock (SYSCLK) selected as USART2 clock
10: HSI16 clock selected as USART2 clock
11: LSE clock selected as USART2 clock

Bits 1:0 **USART1SEL[1:0]**: USART1 clock source selection

This bit is set and cleared by software to select the USART1 clock source.
00: PCLK selected as USART1 clock
01: System clock (SYSCLK) selected as USART1 clock
10: HSI16 clock selected as USART1 clock
11: LSE clock selected as USART1 clock

7.4.27 RTC domain control register (RCC_BDCR)

Address offset: 0x90

Reset value: 0x0000 0000

Reset by RTC domain Reset, except LSCOSEL, LSCOEN and BDRST which are reset only by RTC domain power-on reset.
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 the RTC domain control register (RCC_BDCR) are outside of the V_{CORE} domain. As a result, after Reset, these bits are write-protected and the DBP bit in the Section 6.4.1: Power control register 1 (PWR_CR1) has to be set before these can be modified. Refer to Section 6.1.3: Battery backup domain on page 233 for further information. These bits (except LSCOSEL, LSCOEN and BDRST) are only reset after a RTC domain Reset (see Section 7.1.3: RTC domain reset). Any internal or external Reset will not have any 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.	LSCO SEL	LSCO EN	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**
RTC EN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	LSE CSSD	LSE CSSON	LSEDRV[1:0]		LSE BYP	LSE RDY	LSEON
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

Set and cleared by software.
0: LSI clock selected
1: LSE clock selected

Bit 24 **LSCOEN**: Low speed clock output enable

Set and cleared by software.
0: Low speed clock output (LSCO) disable
1: Low speed clock output (LSCO) enable

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **BDRST**: RTC domain software reset

Set and cleared by software.
0: Reset not activated
1: Reset the entire RTC domain

Bit 15 **RTCEN**: RTC clock enable

Set and cleared by software.
0: RTC clock disabled
1: RTC clock enabled

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**: RTC clock source selection

Set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the RTC domain is reset, or unless a failure is detected on LSE (LSECSSD is set). The BDRST bit can be used to reset them.
00: No clock
01: LSE oscillator clock used as RTC clock
10: LSI oscillator clock used as RTC clock
11: HSE oscillator clock divided by 32 used as RTC clock

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LSECSSD:** CSS on LSE failure Detection

Set by hardware to indicate when a failure has been detected by the Clock Security System on the external 32 kHz oscillator (LSE).
0: No failure detected on LSE (32 kHz oscillator)
1: Failure detected on LSE (32 kHz oscillator)

Bit 5 **LSECSSON:** CSS on LSE enable

Set by software to enable the Clock Security System on LSE (32 kHz oscillator).
LSECSSON must be enabled after the LSE oscillator is enabled (LSEON bit enabled) and ready (LSERDY flag set by hardware), and after the RTCSEL bit is selected.
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 (32 kHz external oscillator) OFF
1: CSS on LSE (32 kHz external oscillator) ON

Bits 4:3 **LSEDRV[1:0]:** LSE oscillator drive capability

Set by software to modulate the LSE oscillator's 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
The oscillator is in Xtal mode when it is not in bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and cleared by software to bypass oscillator in debug mode. This bit 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

Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.
0: LSE oscillator not ready
1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enable

Set and cleared by software.
0: LSE oscillator OFF
1: LSE oscillator ON

7.4.28 Control/status register (RCC_CSR)

Address: 0x94

Reset value: 0x0C00 0000

Reset by system Reset, except reset flags by power Reset only.

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	OBL RSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r		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.	LSI RDY	LSION
														r	rw

Bit 31 **LPWRRSTF**: Low-power reset flag

Set by hardware when a reset occurs due to illegal Stop, Standby or Shutdown mode entry.
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

Set by hardware when a window watchdog reset occurs.
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

Set by hardware when an independent watchdog reset domain occurs.
Cleared by writing to the RMVF bit.

0: No independent watchdog reset occurred
1: Independent watchdog reset occurred

Bit 28 **SFTRSTF**: Software reset flag

Set by hardware when a software reset occurs.
Cleared by writing to the RMVF bit.

0: No software reset occurred
1: Software reset occurred

Bit 27 **BORRSTF**: BOR flag

Set by hardware when a BOR occurs.
Cleared by writing to the RMVF bit.

0: No BOR occurred
1: BOR occurred

Bit 26 **PINRSTF**: Pin reset flag

Set by hardware when a reset from the NRST pin occurs.
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

Set by hardware when a reset from the Option Byte loading occurs.
Cleared by writing to the RMVF bit.

0: No reset from Option Byte loading occurred
1: Reset from Option Byte loading occurred

Bit 24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: Remove reset flag

Set by software to clear the reset flags.

0: No effect
1: Clear the reset flags

Bits 22:2 Reserved, must be kept at reset value.

Bit 1 **LSIRDY**: LSI oscillator ready

Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 LSI oscillator clock cycles. This bit can be set even if LSION = 0 if the LSI is requested by the Clock Security System 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

Set and cleared by software.

0: LSI oscillator OFF

1: LSI oscillator ON

7.4.29 Clock recovery RC register (RCC_CRRCR)

Address: 0x98

Reset value: 0x0000 XXX0

Where X 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.	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**
HSI48CAL[8:0]									Res.	Res.	Res.	Res.	Res.	HSI48 RDY	HSI48 ON
r	r	r	r	r	r	r	r	r						r	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:7 **HSI48CAL[8:0]**: HSI48 clock calibration

These bits are initialized at startup with the factory-programmed HSI48 calibration trim value. They are ready only.

Bits 6:2 Reserved, must be kept at reset value.

Bit 1 **HSI48RDY**: HSI48 clock ready flag

Set by hardware to indicate that HSI48 oscillator is stable. This bit is set only when HSI48 is enabled by software by setting HSI48ON.

0: HSI48 oscillator not ready

1: HSI48 oscillator ready

Bit 0 **HSI48ON**: HSI48 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSI48 when entering in Stop, Standby or Shutdown modes.

0: HSI48 oscillator OFF

1: HSI48 oscillator ON

7.4.30 Peripherals independent clock configuration register (RCC_CCIPR2)

Address: 0x9C

Reset value: 0x0000 0000

Access: no wait state, 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPISEL [1:0]		Res.	Res.	Res.	Res.
										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.	I2C4SEL[1:0]	
														rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **QSPISEL[1:0]:** QUADSPI clock source selection

Set and reset by software.

00: system clock selected as QUADSPI kernel clock
01: HSI16 clock selected as QUADSPI kernel clock
10: PLL “Q” clock selected as QUADSPI kernel clock
11: reserved

Bits 19:2 Reserved, must be kept at reset value.

Bits 1:0 **I2C4SEL[1:0]**: I2C4 clock source selection

These bits are set and cleared by software to select the I2C4 clock source.

00: PCLK selected as I2C4 clock
01: System clock (SYSCLK) selected as I2C4 clock
10: HSI16 clock selected as I2C4 clock
11: reserved

7.4.31 RCC register map

The following table gives the RCC register map and the reset values.

Table 51. RCC 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	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	PLLRDY	PLLON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON	Res.	Res.	Res.	Res.	Res.	HSIRDY	HSIKERON	HSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0	0	0	0	0					0	0	0	0						0	0	0								
0x04	RCC_ICSCR	Res.	HSITRIM[6:0]							HSICAL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		1	0	0	0	0	0	0	X	X	X	X	X	X	X	X																
0x08	RCC_CFGR	Res.	MCOPRE [2:0]			MCOSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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											0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51. RCC 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
0x0C	RCC_PLL CFGR	PLLPDIV[4:0]					PLLR [1:0]		PLLREN	Res.	PLLQ [1:0]		PLLQEN	Res.	Res.	PLLP	PLLPEN	Res.	PLLN [6:0]							PLLM [3:0]				Res.	Res.	PLL SRC [1:0]	
	Reset value	0	0	0	0	0	0	0	0		0	0	0			0	0		0	0	1	0	0	0	0	0	0	0	0			0	0
0x18	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48RDYIE	LSECSSIE	Res.	Res.	Res.	PLLRDYIE	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSIRDYIE
	Reset value																						0	0				0	0	0	0	0	0
0x1C	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48RDYF	LSECSSF	CSSF	Res.	Res.	PLLRDYF	HSERDYF	HSIRDYF	Res.	LSERDYF	LSIRDYF
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x20	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48RDYC	LSECSSC	CSSC	Res.	Res.	PLLRDYC	HSERDYC	HSIRDYC	Res.	LSERDYC	LSIRDYC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x28	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.	FLASHRST	Res.	Res.	Res.	FMACRST	CORDICRST	DMAMUX1RST	DMA2RST	DMA1RST
	Reset value																				0				0				0	0	0	0	0
0x2C	RCC_ AHB2RSTR	Res.	Res.	Res.	Res.	Res.	RNGRST	Res.	AESRST	Res.	Res.	Res.	Res.	DAC4RST	DAC3RST	DAC2RST	DAC1RST	Res.	ADC345RST	ADC12RST	Res.	Res.	Res.	Res.	Res.	Res.	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value						0		0					0	0	0	0		0	0							0	0	0	0	0	0	0
0x30	RCC_ AHB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPIRS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCRST
	Reset value																								0								0
0x38	RCC_ APB1RSTR1	LPTIM1RST	I2C3RST	Res.	PWRRST	Res.	Res.	FDCANRST	Res.	UCPD1RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	Res.	Res.	Res.	Res.	CRSRST	Res.	Res.	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value	0	0		0			0		0	0	0	0	0	0	0		0	0						0			0	0	0	0	0	0
0x3C	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.	UCPD1RST	Res.	Res.	Res.	Res.	Res.	Res.	I2C4RST	LPUART1RST
	Reset value																								0							0	0

Table 51. RCC 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	RCC_ APB2RSTR	Res.	Res.	Res.	Res.	Res.	HRTIMRST	Res.	Res.	Res.	Res.	SAI1RST	TIM20RST	Res.	TIM17RST	TIM16RST	TIM15RST	SPI4RST	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGRST
	Reset value						0					0	0		0	0	0	0	0	0	0	0											0
0x48	RCC_AHB1 ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	FMACEN	CORDICEN	DMAMUX1EN	DMA2EN	DMA1EN
	Reset value																				0				1				0	0	0	0	0
0x4C	RCC_AHB2 ENR	Res.	Res.	Res.	Res.	Res.	RNGEN	Res.	AESEN	Res.	Res.	Res.	Res.	DAC4EN	DAC3EN	DAC2EN	DAC1EN	Res.	ADC345EN	ADC12EN	Res.	Res.	Res.	Res.	Res.	Res.	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value						0		0					0	0	0	0		0	0							0	0	0	0	0	0	0
0x50	RCC_AHB3 ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCEN
	Reset value																								0								0
0x58	RCC_ APB1ENR1	LPTIM1EN	I2C3EN	Res.	PWREN	Res.	Res.	FDCANEN	Res.	UCPD1EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SP3EN	SPI2EN	Res.	Res.	WWDGEN	RTCAPBEN	Res.	CRSEN	Res.	Res.	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0	0		0			0		0	0	0	0	0	0	0		0	0			0	1		0			0	0	0	0	0	0
0x5C	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.	UCPD1EN	Res.	Res.	Res.	Res.	Res.	Res.	I2C4EN	LPUART1EN
	Reset value																								0							0	0
0x60	RCC_ APB2ENR	Res.	Res.	Res.	Res.	Res.	HRTIM1EN	Res.	Res.	Res.	Res.	SAI1EN	TIM20EN	Res.	TIM17EN	TIM16EN	TIM15EN	SPI4EN	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	Reset value						0					0	0		0	0	0	0	0	0	0	0											0
0x68	RCC_ AHB1SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCSMEN	Res.	Res.	SRAM1SMEN	FLASHSMEN	Res.	Res.	Res.	FMACSMEN	CORDICSMEN	DMAMUX1SMEN	DMA2SMEN	DMA1SMEN
	Reset value																				1			1	1				1	1	1	1	1

Table 51. RCC 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
0x6C	RCC_ AHB2SMENR	Res.	Res.	Res.	Res.	Res.	RNGSMEN	Res.	AESSMEN	Res.	Res.	Res.	Res.	DAC4SMEN	DAC3SMEN	DAC2SMEN	DAC1SMEN	Res.	ADC345SMEN	ADC12SMEN	Res.	Res.	CCMSRAMSMEN	SRAM2SMEN	Res.	Res.	GPIOGSMEN	GPIOFSMEN	GPIOESMEN	GPIODSMEN	GPIOCSMEN	GPIOBSMEN	GPIOASMEN
	Reset value						1		1					1	1	1	1		1	1			1	1			1	1	1	1	1	1	1
0x70	RCC_ AHB3SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPISMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCSMEN
	Reset value																								1								1
0x78	RCC_ APB1SM ENR1	LPTIM1SMEN	I2C3SMEN	Res.	PWRSMEN	Res.	Res.	FDCANSMEN	Res.	UCPD1SMEN	I2C2SMEN	I2C1SMEN	UART5SMEN	UART4SMEN	USART3SMEN	USART2SMEN	Res.	SP3SMEN	SPI2SMEN	Res.	Res.	WWDGSMEN	RTCAPBSMEN	Res.	CRSSMEN	Res.	Res.	TIM7SMEN	TIM6SMEN	TIM5SMEN	TIM4SMEN	TIM3SMEN	TIM2SMEN
	Reset value	1	1		1			1		1	1	1	1	1	1	1		1	1			1	1		1			1	1	1	1	1	1
0x7C	RCC_ APB1SM ENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	I2C4SMEN	LPUART1SMEN
	Reset value																								0							1	1
0x80	RCC_ APB2SMENR	Res.	Res.	Res.	Res.	Res.	HRTIM1SMEN	Res.	Res.	Res.	Res.	SAI1SMEN	TIM20SMEN	Res.	TIM17SMEN	TIM16SMEN	TIM15SMEN	Res.	USART1SMEN	TIM8SMEN	SPI1SMEN	TIM1SMEN	SPI4SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGSMEN
	Reset value						1					1	1		1	1	1		1	1	1	1	0										1
0x88	RCC_CCIPR	ADC345SEL [1:0]		ADC12SEL [1:0]		CLK48SEL [1:0]		FDCANSEL [1:0]		I2S23SEL [1:0]		SAI1SEL [1:0]		LPTIM1SEL [1:0]		I2C3SEL [1:0]		I2C2SEL [1:0]		I2C1SEL [1:0]		LPUART1SEL [1:0]		UART5SEL [1:0]		UART4SEL [1:0]		USART3SEL [1:0]		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	0	0	0	0	0	0	0	0
0x90	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL [1:0]		Res.	LSECSSD	LSECSSON	LSE DRV [1:0]		LSEBYP	LSERDY	LSEON
	Reset value							0	0								0	0						0	0		0	0	0	0	0	0	0
0x94	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset value	0	0	0	0	1	1	0		0																						0	0

Table 51. RCC 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
0x98	RCC_CRRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[8:0]									Res.	Res.	Res.	Res.	Res.	HSI48RDY	HSI48ON
	Reset value																	X	X	X	X	X	X	X	X	X						0	0
0x9C	RCC_CCIPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPISEL		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C4SEL [1:0]	
	Reset value											0	0																			0	0

Refer to *Section 2.2 on page 81* for the register boundary addresses.

8 Clock recovery system (CRS)

8.1 Introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means for oscillator output frequency evaluation, based on comparison with a selectable synchronization signal. It is capable of doing automatic adjustment of oscillator trimming based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In such case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, which is sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from the LSE oscillator output or it can be generated by user software.

8.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - LSE oscillator output
  - packet reception
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster start-up convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

8.3 CRS implementation

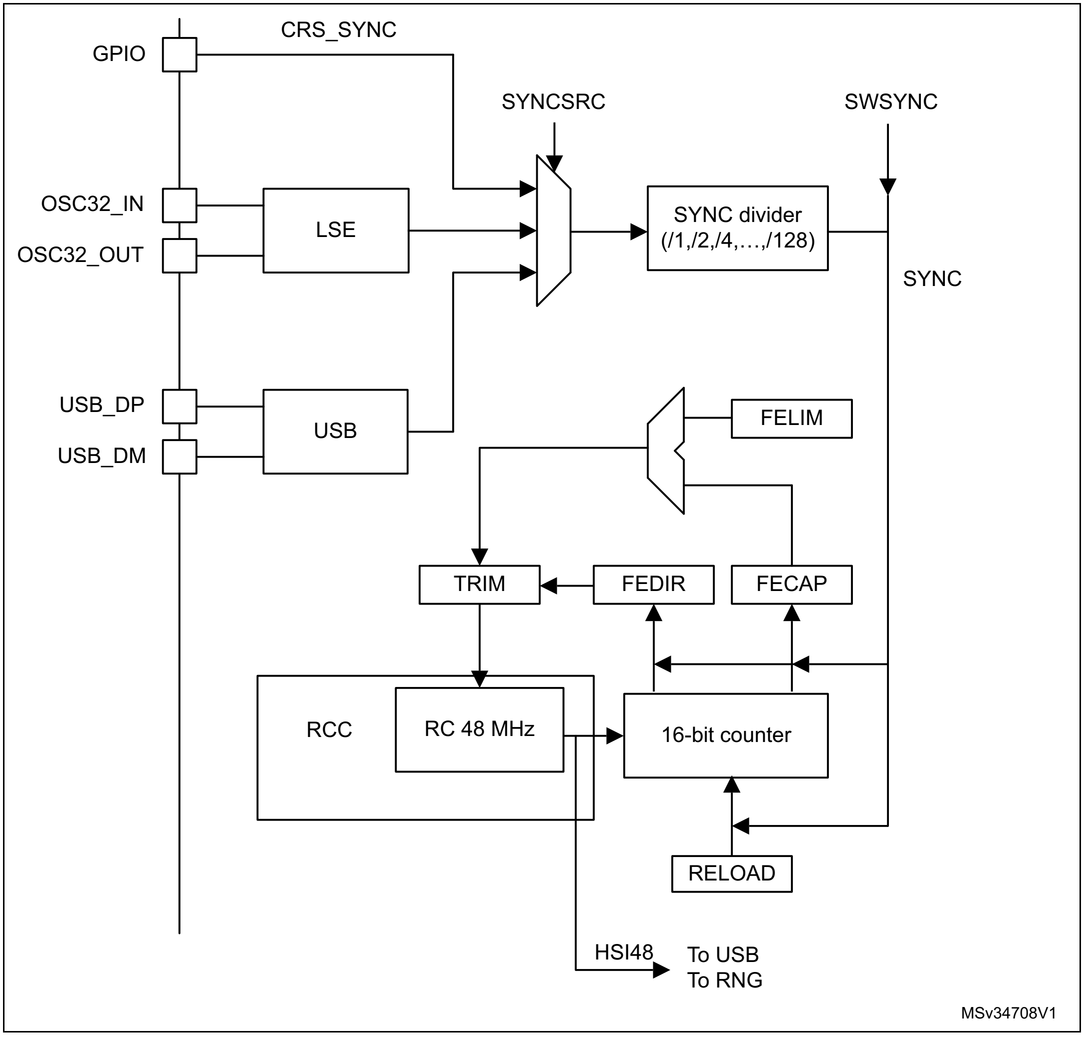
Table 52. CRS features

Feature	CRS1
TRIM width	7 bits

8.4 CRS functional description

8.4.1 CRS block diagram

Figure 23. CRS block diagram



8.4.2 Synchronization input

The CRS synchronization (SYNC) source, selectable through the CRS_CFGR register, can be the signal from the LSE clock or the USB SOF signal. For a better robustness of the SYNC input, a simple digital filter (2 out of 3 majority votes, sampled by the RC48 clock) is implemented to filter out any glitches. This source signal also has a configurable polarity and can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to *Section 8.7.2: CRS configuration register (CRS_CFGR)*.

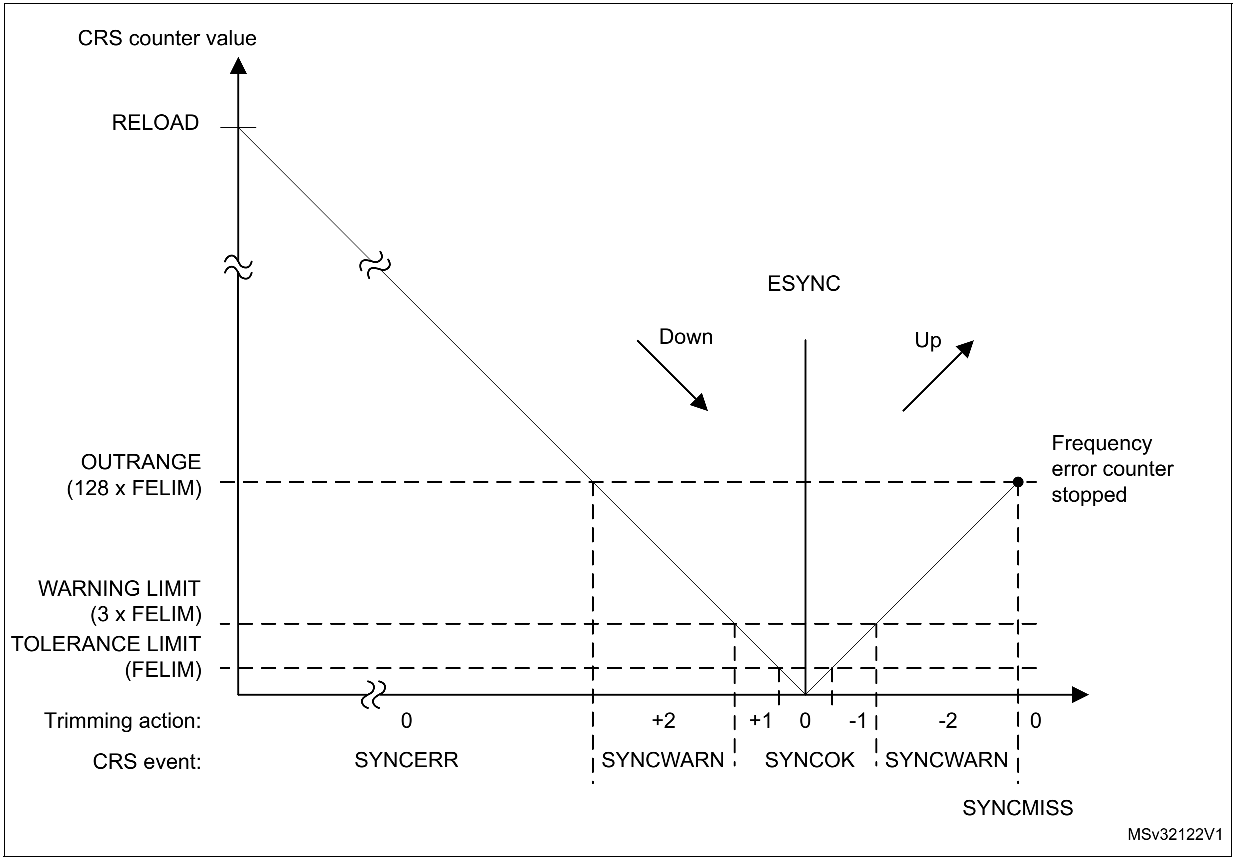
It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

8.4.3 Frequency error measurement

The frequency error counter is a 16-bit down/up counter which is reloaded with the RELOAD value on each SYNC event. It starts counting down till it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit where it eventually stops (if no SYNC event is received) and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the downcounting phase (before reaching the zero value), it means that the actual frequency is lower than the target (and so, that the TRIM value must be incremented), while when it is detected during the upcounting phase it means that the actual frequency is higher (and that the TRIM value must be decremented).

Figure 24. CRS counter behavior



8.4.4 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 * FELIM value
- OUTRANGE (error limit), defined as 128 * FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated
  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, it means that a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, it means that the frequency is out of the trimming range. This can also happen when the SYNC input is not clean or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

Note: *If the actual value of the TRIM field is so close to its limits that the automatic trimming would force it to overflow or underflow, then the TRIM value is set just to the limit and the TRIMOVF status is indicated.*

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

8.4.5 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. It is then decreased by one to reach the expected synchronization on the zero value. The formula is the following:

$$\text{RELOAD} = (f_{\text{TARGET}} / f_{\text{SYNC}}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of HSI48 oscillator clock ticks. The following formula can be used:

$$\text{FELIM} = (f_{\text{TARGET}} / f_{\text{SYNC}}) * \text{STEP}[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to (f_{TARGET} / f_{SYNC}) = 48000 and to a typical trimming step size of 0.14%.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields which can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

8.5 CRS low-power modes

Table 53. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator restarted.
Standby	The CRS peripheral is powered down and must be reinitialized after exiting Standby mode.

8.6 CRS interrupts

Table 54. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

8.7 CRS registers

Refer to *Section 1.2 on page 73* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

8.7.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 X000 (X=4 for products supporting 7-bit TRIM width, otherwise X=2)

31	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.	TRIM[6:0]							SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNC WARNIE	SYNC OKIE
	rw	rw	rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TRIM[6:0]**: HSI48 oscillator smooth trimming

For product supporting the 7-bit TRIM width (see *Section 8.3*), the default value of the HSI48 oscillator smooth trimming is 64, which corresponds to the middle of the trimming interval.

For products supporting the 6-bit TRIM width (see *Section 8.3*) this bit is reserved, must be kept at reset value.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 8.4.4* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable

0: Synchronization or trimming error (ERRF) interrupt disabled
1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable

0: SYNC warning (SYNCWARNF) interrupt disabled
1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable

0: SYNC event OK (SYNCOKF) interrupt disabled
1: SYNC event OK (SYNCOKF) interrupt enabled

8.7.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYNCPOL	Res.	SYNCSRC[1:0]		Res.	SYNCDIV[2:0]			FELIM[7:0]							
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
RELOAD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.
0: SYNC active on rising edge (default)
1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source.
00: GPIO selected as SYNC signal source
01: LSE selected as SYNC signal source
10: USB SOF selected as SYNC signal source (default).
11: Reserved

Note: When using USB LPM (Link Power Management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the HSI48 oscillator on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE or reference clock on the GPIOs should be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.
000: SYNC not divided (default)
001: SYNC divided by 2
010: SYNC divided by 4
011: SYNC divided by 8
100: SYNC divided by 16
101: SYNC divided by 32
110: SYNC divided by 64
111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to *Section 8.4.4* for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to *Section 8.4.3* for more details about counter behavior.

8.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[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
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to *Section 8.4.4* for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.
0: Upcounting direction, the actual frequency is above the target.
1: Downcounting direction, the actual frequency is below the target.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.
0: No trimming error signalized
1: Trimming error signalized

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reached value FELIM * 128 and no SYNC was detected, meaning either that a SYNC pulse was missed or that the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, and that some other action has to be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC) and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC missed error signalized

1: SYNC missed error signalized

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC error signalized

1: SYNC error signalized

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.

0: No expected SYNC signalized

1: Expected SYNC signalized

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signalized

1: Synchronization or trimming error signalized

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.

0: No SYNC warning signalized

1: SYNC warning signalized

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.

0: No SYNC event OK signalized

1: SYNC event OK signalized

8.7.4 CRS interrupt flag clear register (CRS_ICR)

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**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNC WARNC	SYNC OKC
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCC**: Expected SYNC clear flag

Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag

Writing 1 to this bit clears TRIMOVF, SYNCMISS and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag

Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag

Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

8.7.5 CRS register map

Table 55. CRS 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	**CRS_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[6]	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value																		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**CRS_CFGR**	SYNCPOL	Res.	SYNC SRC [1:0]		Res.	SYNC DIV [2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	**CRS_ISR**	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	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
0x0C	**CRS_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

Refer to *Section 2.2 on page 81* for the register boundary addresses.

9 General-purpose I/Os (GPIO)

9.1 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 a 32-bit set/reset register (GPIOx_BSRR). In addition all GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

9.2 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 allows the use of I/O pins as GPIOs or as one of several peripheral functions

9.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- 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 have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR register is to 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 25 and *Figure 26* show the basic structures of a standard and a 5-Volt tolerant I/O port bit, respectively. *Table 56* gives the possible port bit configurations.

Figure 25. Basic structure of an I/O port bit

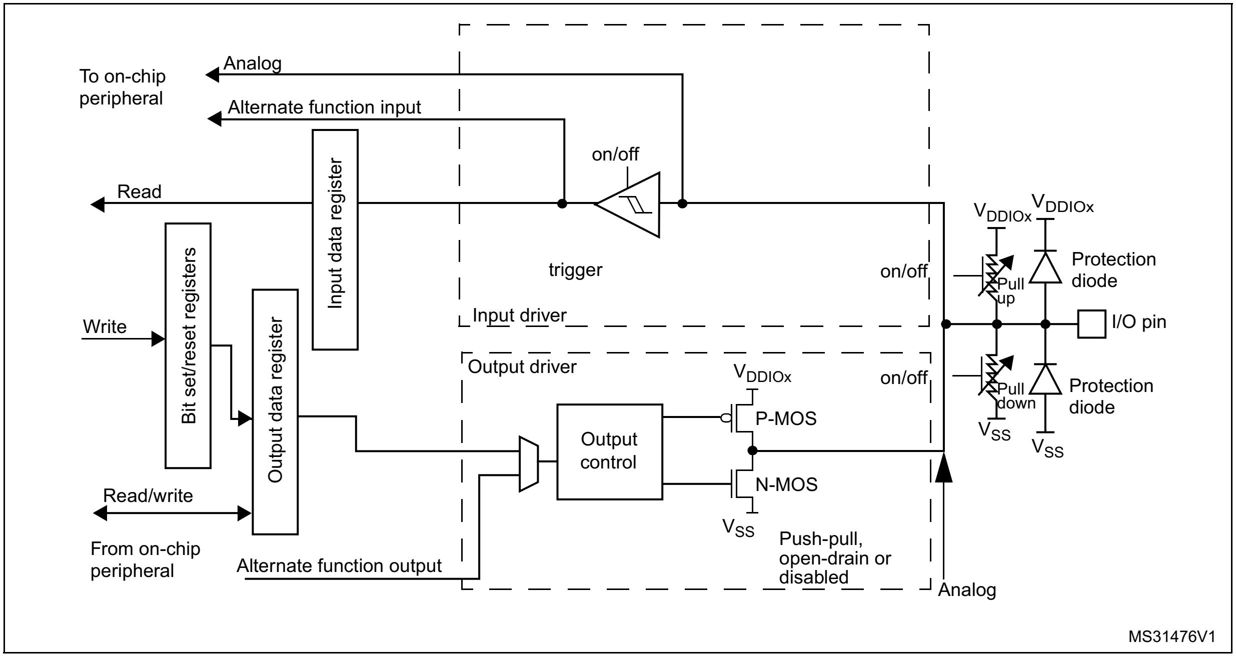
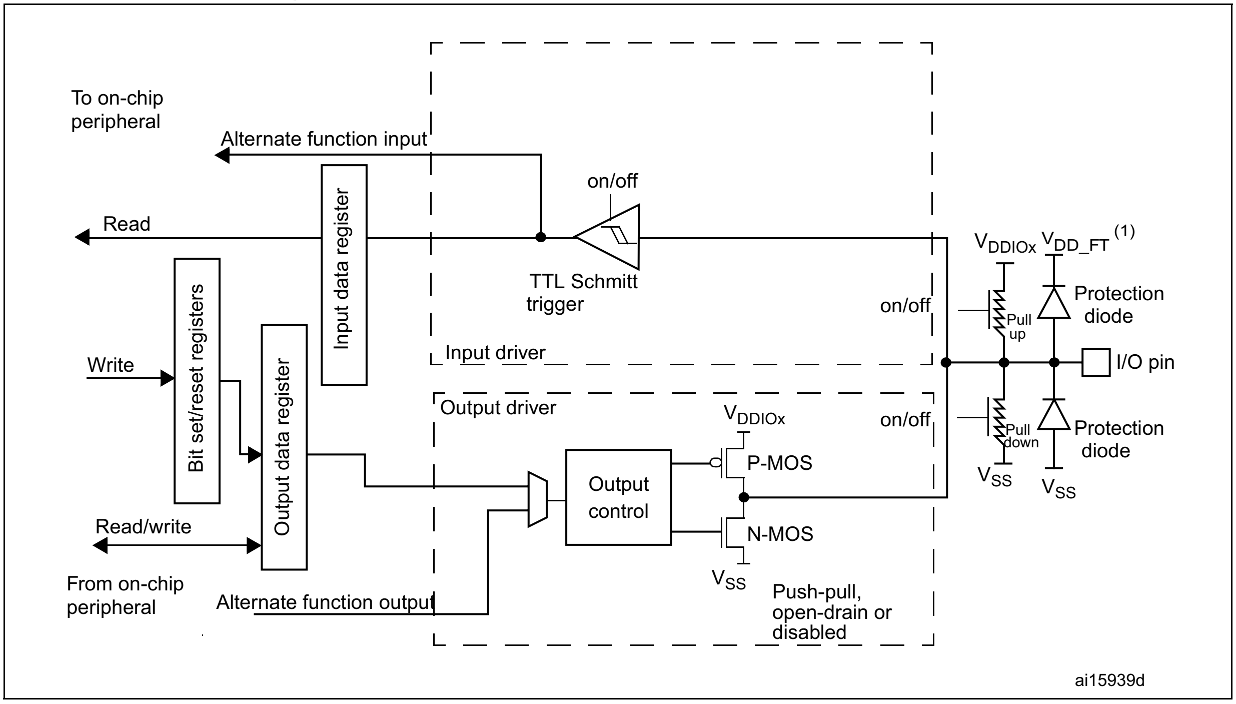


Figure 26. Basic structure of a 5-Volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Table 56. Port bit configuration table[1]

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/output	Analog, PD
	x	x	x	1	1	Reserved	

[1] GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

9.3.1 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 are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDAT in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO in floating state no pull-up/pull-down

PB8/BOOT0 is in input mode during the reset until at least the end of the option byte loading phase. See *Section 9.3.15: Using PB8 as GPIO*.

When the 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.

9.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no 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 GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto 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 has to 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 one of 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_OSPEEDER registers, respectively.

  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions:**
  - For the ADC, DAC, OPAMP and COMP, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC, DAC, OPAMP, and COMP registers.

    As indicated above, for the additional functions (such as DAC or OPAMP), the output is controlled by the corresponding peripheral. Care must be taken to select the I/O port analog function before enabling the additional function output in the peripheral control register.
  - 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 device datasheet for the detailed mapping of the alternate function I/O pins.

9.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

9.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

See *Section 9.4.5: GPIO port input data register (GPIOx_IDR) (x = A to G)* and *Section 9.4.6: GPIO port output data register (GPIOx_ODR) (x = A to G)* for the register descriptions.

9.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register 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, bit BS(i) **sets** the corresponding ODR(i) bit. When written to 1, bit 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 the GPIOx_BSRR register 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. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

9.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to 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 configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been 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 (refer to *Section 9.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to G)*) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in *Section 9.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to G)*.

9.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. 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 using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. 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 device datasheet.

9.3.8 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 15: Extended interrupts and events controller (EXTI)* and to *Section 15.3.2: Wakeup event management*.

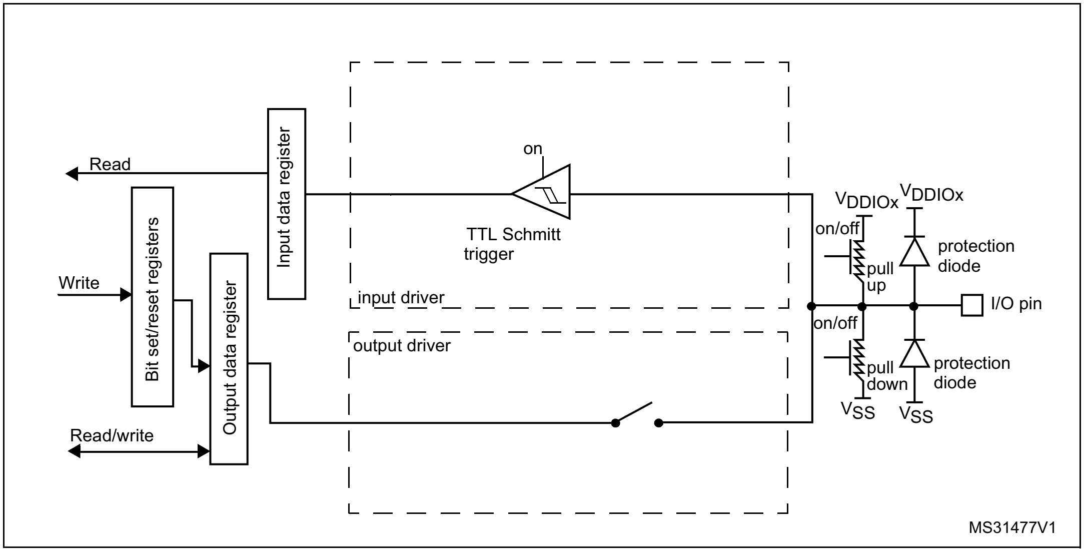
9.3.9 Input configuration

When the I/O port is programmed as input:

- 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

Figure 27 shows the input configuration of the I/O port bit.

Figure 27. Input floating/pull up/pull down configurations



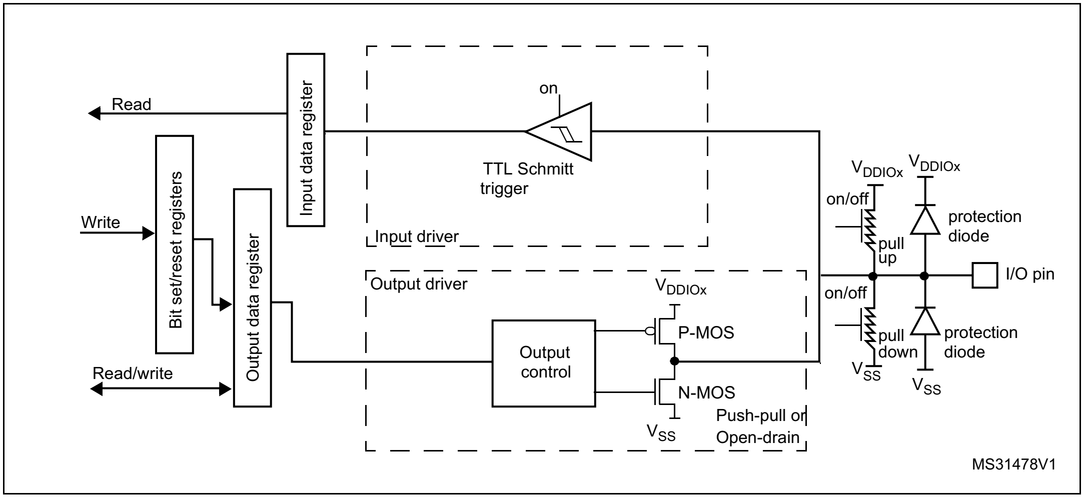
9.3.10 Output configuration

When the I/O port is programmed as output:

- 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

Figure 28 shows the output configuration of the I/O port bit.

Figure 28. Output configuration



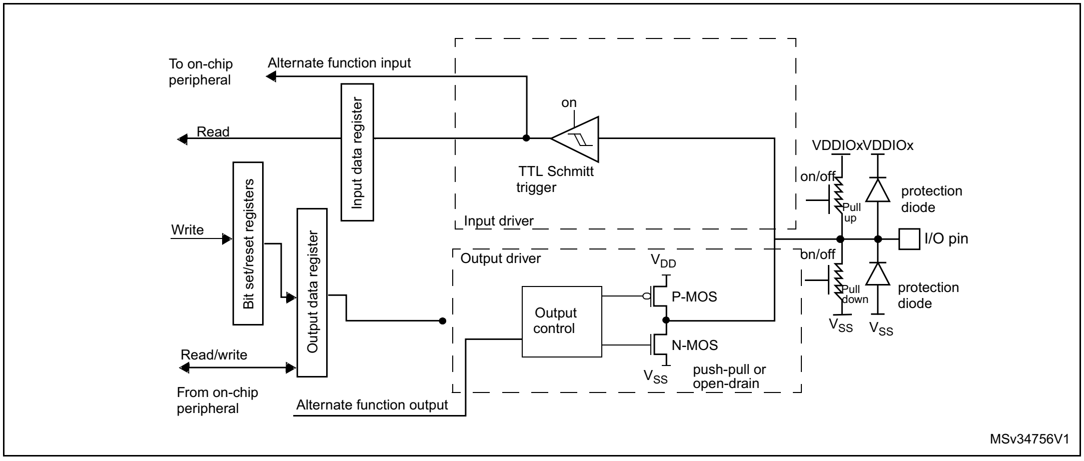
9.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- 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

Figure 29 shows the Alternate function configuration of the I/O port bit.

Figure 29. Alternate function configuration



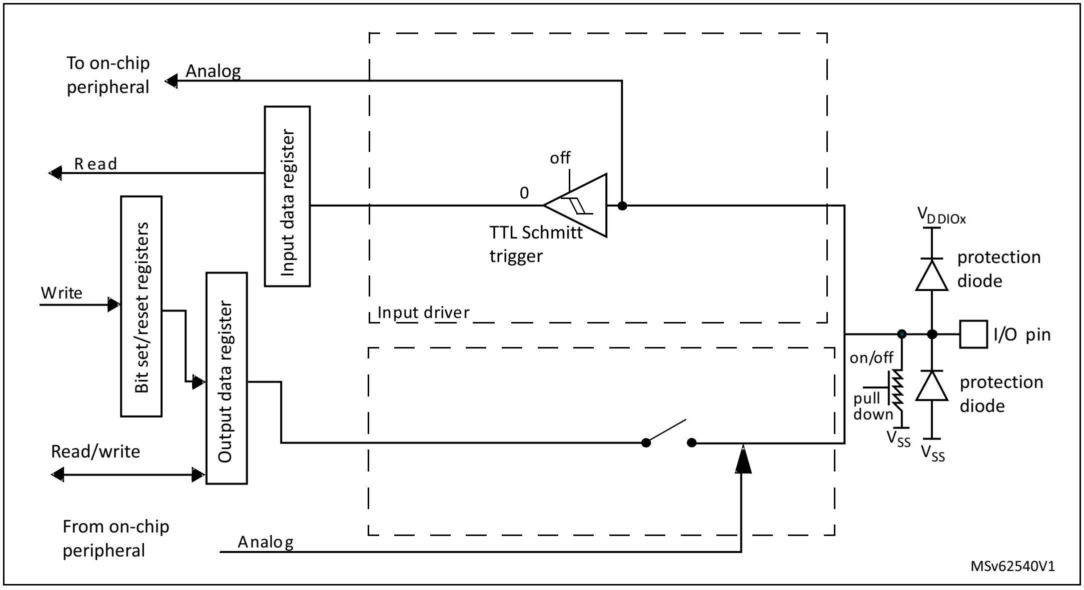
9.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- 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 is disabled by Hardware. The weak pull-down is configurable.
- Read access to the input data register gets the value “0”

Figure 30 shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 30. High impedance-analog configuration



9.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the HSEON or LSEON bit in the RCC_CSR register) the oscillator takes 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 OSC_IN or OSC32_IN pin is reserved for clock input and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

9.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the 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 35.3: RTC functional description*.

9.3.15 Using PB8 as GPIO

PB8 may be used as boot pin (BOOT0) or as a GPIO. Depending on the nSWBOOT0 bit in the user option byte, it switches from the input mode to the analog input mode:

- After the option byte loading phase if nSWBOOT0 = 1.
- After reset if nSWBOOT0 = 0.

9.3.16 Using PG10 as GPIO

PG10 may be used as reset pin (NRST) or as a GPIO. Depending on the NRST_MODE bits in the user option byte, it switches to those mode:

- Reset input/output: default at power-on reset or after option bytes loading NRST_MODE = 3
- Reset input only: after option bytes loading NRST_MODE = 1
- GPIO PG10 mode: after option bytes loading NRST_MODE = 2

See description on the NRST pin in *Section 7.1.2: System reset*

9.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to *Table 57*.

The peripheral registers can be written in word, half word or byte mode.

9.4.1 GPIO port mode register (GPIOx_MODER) (x =A to G)

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFFFF FEBF (for port B)

Reset value: 0xFFFF FFFF (for ports C..G)

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 **MODE[15:0][1:0]:** Port x configuration I/O pin y (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)

9.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to G)

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
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 **OT[15:0]:** Port x configuration I/O pin y (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

9.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to G)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 0000 (for the other ports)

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 **OSPEED[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very high speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed..

9.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to G)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for other ports)

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 **PUPD[15:0][1:0]:** Port x configuration I/O pin y (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

9.4.5 GPIO port input data register (GPIOx_IDR) (x = A to G)

Address offset: 0x10

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 **ID[15:0]:** Port x input data I/O pin y (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.

9.4.6 GPIO port output data register (GPIOx_ODR) (x = A to G)

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.	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 **OD[15:0]:** Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR register (x = A..F).

9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to G)

Address offset: 0x18

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
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**
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit
1: Resets the corresponding ODx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BS[15:0]:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit
1: Sets the corresponding ODx bit

9.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to G)

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 GPIOx_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).

Address offset: 0x1C

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.	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 **LCK[15:0]:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is ‘0.

0: Port configuration not locked

1: Port configuration locked

9.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to G)

Address offset: 0x20

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 **AFSEL[7:0][3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)

These bits are written by software to configure alternate function I/Os.

0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

9.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to G)

Address offset: 0x24

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 **AFSEL[15:8][3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)

These bits are written by software to configure alternate function I/Os.

0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

9.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to G)

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**
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
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 **BR[15:0]:** Port x reset IO pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODx bit
1: Reset the corresponding ODx bit

9.4.12 GPIO register map

The following table gives the GPIO register map and reset values.

Table 57. GPIO 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	**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
0x00	**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
0x00	**GPIOx_MODER** (where x = **C..G**)	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	1	1	1	1	1	1	1	1	1
0x04	**GPIOx_OTYPER** (where x = **A..G**)	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
0x08	**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
0x08	**GPIOx_OSPEEDR** (where x = **A to G**)	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	0	0	0	0	0	0	0	0
0x0C	**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
0x0C	**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
0x0C	**GPIOx_PUPDR** (where x =)	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	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = **A to G**)	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

Table 57. GPIO 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
0x14	**GPIOx_ODR** (where x = **A to G**)	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
0x18	**GPIOx_BSRR** (where x = **A to G**)	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
0x1C	**GPIOx_LCKR** (where x = **A to G**)	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
0x20	**GPIOx_AFRL** (where x = **A to G**)	7[3:0]				6[3:0]				5[3:0]				4[3:0]				3[3:0]				2[3:0]				1[3:0]				0[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
0x24	**GPIOx_AFRH** (where x = **A to G**)	15[3:0]				14[3:0]				13[3:0]				12[3:0]				11[3:0]				10[3:0]				9[3:0]				8[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
0x28	**GPIOx_BRR** (where x = **A to G**))	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 2.2 on page 81* for the register boundary addresses.

10 System configuration controller (SYSCFG)

10.1 SYSCFG main features

The STM32G4 Series 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 CCM RAM write protection and software erase
- Configuring FPU interrupts
- Enabling /disabling I^2C Fast-mode Plus driving capability on some I/Os and voltage booster for I/Os analog switches.

10.2 SYSCFG registers

10.2.1 SYSCFG memory remap register (SYSCFG_MEMRMP)

This register is used for specific configurations on memory remap.

Address offset: 0x00

Reset value: 0x0000 000X (X 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	FB_ MODE	Res	Res	Res	Res	Res	MEM_MODE		
							rw						rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **FB_MODE:** Flash Bank mode selection

0: Flash Bank 1 mapped at 0x0800 0000 (and aliased @0x0000 0000[1]) and Flash Bank 2 mapped at 0x0804 0000 (and aliased at 0x0008 0000)

1: Flash Bank2 mapped at 0x0800 0000 (and aliased @0x0000 0000[1]) and Flash Bank 1 mapped at 0x0804 0000 (and aliased at 0x0008 0000)

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

Bits 2:0 **MEM_MODE:** 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 pin and the option bit setting. After reset these bits take the value selected by BOOT0 pin (or option bitnSWBOOT0) and BOOT1 option bit.

000: Main Flash memory mapped at 0x00000000[1].
001: System Flash memory mapped at 0x00000000.
010: FSMC memory.
011: SRAM1 mapped at 0x00000000.
100: QUADSPI memory mapped at 0x0000 0000
101: Reserved
111: Reserved

[1] When BFB2 bit is set, the system memory remains aliased at @0x0000 0000

Note: When the FSMC is remapped at address 0x0000 0000, only the first two regions of Bank 1 memory controller (Bank1 NOR/PSRAM 1 and NOR/PSRAM 2) can be remapped. In remap mode, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.

10.2.2 SYSCFG configuration register 1 (SYSCFG_CFGR1)

Address offset: 0x04

Reset value: 0x7C00 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPU_IE[5..0]						Res	Res	I2C4_ FMP	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	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	ANAS WVDD	BOOST EN	Res	Res	Res	Res	Res	Res	Res	Res
						rw	rw								

Bits 31:26 **FPU_IE[5..0]**: Floating Point Unit interrupts enable bits
FPU_IE[5]: Inexact interrupt enable
FPU_IE[4]: Input denormal interrupt enable
FPU_IE[3]: Overflow interrupt enable
FPU_IE[2]: underflow interrupt enable
FPU_IE[1]: Divide-by-zero interrupt enable
FPU_IE[0]: Invalid operation interrupt enable

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **I2C4_FMP**: Fast-mode Plus driving capability activation

This bit enables the Fm+ driving mode on I2C4 pins selected through AF selection bits.
0: Fm+ mode is not enabled on I2C4 pins selected through AF selection bits
1: Fm+ mode is enabled on I2C4 pins selected through AF selection bits.

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

Bit 9 **ANASWVDD:** GPIO analog switch control voltage selection
0: I/O analog switches supplied by VDDA or booster when booster is ON
1: I/O analog switches supplied by VDD.
Refer to *Table 58* for bit 9 setting.

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.

Table 58 describes when the bit 9 (ANASWVDD) and the bit 8 (BOOSTEN) should be set or reset depending on the voltage settings.

Note: When FM+ mode is activated on GPIO pin, the speed configuration of the GPIO (in GPIOx_OSPEEDR register) is ignored. Program this register only after the AF selection through the GPIOx_AFRH or GPIOx_AFRL register.

Table 58. BOOSTEN and ANASWVDD set/reset

VDD	VDDA	BOOSTEN	ANASWVDD
-	> 2.4 V	0	0
> 2.4 V	< 2.4 V	0	1
< 2.4 V	< 2.4 V	1	0

10.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

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**
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
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 **EXTI3[3:0]**: EXTI 3 configuration bits

These bits are written by software to select the source input for the EXTI3 external interrupt.

0000: PA[3] pin
0001: PB[3] pin
0010: PC[3] pin
0011: PD[3] pin
0100: PE[3] pin
0101: PF[3] pin
0110: PG[3] pin

Bits 11:8 **EXTI2[3:0]**: EXTI 2 configuration bits

These bits are written by software to select the source input for the EXTI2 external interrupt.

0000: PA[2] pin
0001: PB[2] pin
0010: PC[2] pin
0011: PD[2] pin
0100: PE[2] pin
0101: PF[2] pin
0110: PG[2] pin

Bits 7:4 **EXTI1[3:0]**: EXTI 1 configuration bits

These bits are written by software to select the source input for the EXTI1 external interrupt.

0000: PA[1] pin
0001: PB[1] pin
0010: PC[1] pin
0011: PD[1] pin
0100: PE[1] pin
0101: PF[1] pin
0110: PG[1] pin

Bits 3:0 **EXTI0[3:0]**: EXTI 0 configuration bits

These bits are written by software to select the source input for the EXTI0 external interrupt.

0000: PA[0] pin
0001: PB[0] pin
0010: PC[0] pin
0011: PD[0] pin
0100: PE[0] pin
0101: PF[0] pin
0110: PG[0] pin

10.2.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

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
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
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 **EXTI7[3:0]**: EXTI 7 configuration bits

These bits are written by software to select the source input for the EXTI7 external interrupt.

0000: PA[7] pin
0001: PB[7] pin
0010: PC[7] pin
0011: PD[7] pin
0100: PE[7] pin
0101: PF[7] pin
0110: PG[7] pin

Bits 11:8 **EXTI6[3:0]**: EXTI 6 configuration bits

These bits are written by software to select the source input for the EXTI6 external interrupt.

0000: PA[6] pin
0001: PB[6] pin
0010: PC[6] pin
0011: PD[6] pin
0100: PE[6] pin
0101: PF[6] pin
0110: PG[6] pin

Bits 7:4 **EXTI5[3:0]**: EXTI 5 configuration bits

These bits are written by software to select the source input for the EXTI5 external interrupt.

0000: PA[5] pin
0001: PB[5] pin
0010: PC[5] pin
0011: PD[5] pin
0100: PE[5] pin
0101: PF[5] pin
0110: PG[5] pin

Bits 3:0 **EXTI4[3:0]**: EXTI 4 configuration bits

These bits are written by software to select the source input for the EXTI4 external interrupt.

0000: PA[4] pin
0001: PB[4] pin
0010: PC[4] pin
0011: PD[4] pin
0100: PE[4] pin
0101: PF[4] pin
0110: PG[4] pin

Note: Some of the I/O pins mentioned in the above register may not be available on small packages.

10.2.5 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)

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
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
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 **EXTI11[3:0]**: EXTI 11 configuration bits

These bits are written by software to select the source input for the EXTI11 external interrupt.

0000: PA[11] pin
0001: PB[11] pin
0010: PC[11] pin
0011: PD[11] pin
0100: PE[11] pin
0101: PF[11] pin

Bits 11:8 **EXTI10[3:0]**: EXTI 10 configuration bits

These bits are written by software to select the source input for the EXTI10 external interrupt.

0000: PA[10] pin
0001: PB[10] pin
0010: PC[10] pin
0011: PD[10] pin
0100: PE[10] pin
0101: PF[10] pin
0110: PG[10] pin

Bits 7:4 **EXTI9[3:0]**: EXTI 9 configuration bits

These bits are written by software to select the source input for the EXTI9 external interrupt.

0000: PA[9] pin
0001: PB[9] pin
0010: PC[9] pin
0011: PD[9] pin
0100: PE[9] pin
0101: PF[9] pin
0110: PG[9] pin

Bits 3:0 **EXTI8[3:0]**: EXTI 8 configuration bits

These bits are written by software to select the source input for the EXTI8 external interrupt.

0000: PA[8] pin
0001: PB[8] pin
0010: PC[8] pin
0011: PD[8] pin
0100: PE[8] pin
0101: PF[8] pin
0110: PG[8] pin

Note: Some of the I/O pins mentioned in the above register may not be available on small packages.

10.2.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)

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	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**
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
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 **EXTI15[3:0]**: EXTI 15 configuration bits

These bits are written by software to select the source input for the EXTI15 external interrupt.

0000: PA[15] pin
0001: PB[15] pin
0010: PC[15] pin
0011: PD[15] pin
0100: PE[15] pin
0101: PF[15] pin

Bits 11:8 **EXTI14[3:0]**: EXTI 14 configuration bits

These bits are written by software to select the source input for the EXTI14 external interrupt.

0000: PA[14] pin
0001: PB[14] pin
0010: PC[14] pin
0011: PD[14] pin
0100: PE[14] pin
0101: PF[14] pin

Bits 7:4 **EXTI13[3:0]**: EXTI 13 configuration bits

These bits are written by software to select the source input for the EXTI13 external interrupt.

0000: PA[13] pin
0001: PB[13] pin
0010: PC[13] pin
0011: PD[13] pin
0100: PE[13] pin
0101: PF[13] pin

Bits 3:0 **EXTI12[3:0]**: EXTI 12 configuration bits

These bits are written by software to select the source input for the EXTI12 external interrupt.

0000: PA[12] pin
0001: PB[12] pin
0010: PC[12] pin
0011: PD[12] pin
0100: PE[12] pin
0101: PF[12] pin

Note: Some of the I/O pins mentioned in the above register may not be available on small packages.

10.2.7 SYSCFG CCM SRAM control and status register (SYSCFG_SCSR)

Address offset: 0x18

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	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	CCM BSY	CCM ER
														r	rw

Bits 31:2 Reserved, must be kept at reset value

Bit 1 **CCMBSY:** CCM SRAM busy by erase operation

0: No CCM SRAM erase operation is on going.

1: CCM SRAM erase operation is on going.

Bit 0 **CCMER:** CCM SRAM Erase

Setting this bit starts a hardware CCM SRAM erase operation. This bit is automatically cleared at the end of the CCM SRAM 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.

10.2.8 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x1C

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	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**: SRAM1 and CCM SRAM parity error flag

This bit is set by hardware when an SRAM1 or CCM SRAM parity error is detected. It is cleared by software by writing ‘1’.

0: No parity error detected

1: 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 TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1.

0: ECC error disconnected from TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1.

1: ECC error connected to TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1.

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 TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1., as well as the PVDE and PLS[2:0] in the PWR_CR2 register.

0: PVD interrupt disconnected from TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1. PVDE and PLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1, PVDE and PLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM1 and CCM SRAM 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 SRAM1 or CCM SRAM parity error signal connection to TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1.

0: CCM SRAM parity error signal disconnected from TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1.

1: CCM SRAM parity error signal connected to TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1.

Bit 0 **CLL:** Cortex$^{®}$-M4 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 Cortex$^{®}$-M4 with FPU LOCKUP (Hardfault) output to TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1.

0: Cortex$^{®}$-M4 LOCKUP output disconnected from TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1.

1: Cortex$^{®}$-M4 LOCKUP output connected to TIM1/8/15/16/17/20 break input and hrtim_sys_flt input of HRTIM1.

10.2.9 SYSCFG CCM SRAM write protection register (SYSCFG_SWPR)

Address offset: 0x20

System 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** (x = 0 to 31): CCM SRAM page x write protection

These bits are set by software and cleared only by a system reset.

0: Write protection of CCM SRAM page x is disabled.

1: Write protection of CCM SRAM page x is enabled.

10.2.10 SYSCFG CCM SRAM key register (SYSCFG_SKR)

Address offset: 0x24

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	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]**: CCM SRAM write protection key for software erase

The following steps are required to unlock the write protection of the CCMER 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.

10.2.11 SYSCFG register map

The following table gives the SYSCFG register map and the reset values.

Table 59. SYSCFG 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	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.	FB_MODE	Res.	Res.	Res.	Res.	Res.	MEM_ MODE		
	Reset value																								0						x	x	x
0x04	SYSCFG_CFGR1	FPU_IE[5..0]						Res.	Res.	I2C4_FMP	I2C3_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP	Res.	Res.	Res.	Res.	Res.	Res.	ANASWVDD	BOOSTEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FWDIS
	Reset value	0	1	1	1	1	1			0	0	0	0	0	0	0	0							0	0					0	0	0	1
0x08	SYSCFG_EXTICR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI3 [3:0]				EXTI2 [3:0]				EXTI1 [3:0]				EXTI0 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_EXTICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI7 [3:0]				EXTI6 [3:0]				EXTI5 [3:0]				EXTI4 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI11 [3:0]				EXTI10 [3:0]				EXTI9 [3:0]				EXTI8 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI15 [3:0]				EXTI14 [3:0]				EXTI13 [3:0]				EXTI12 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMBSY	CCMER
	Reset value																															0	0
0x1C	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
0x20	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
0x24	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							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 81* for the register boundary addresses.

11 Peripherals interconnect matrix

11.1 Introduction

Several peripherals have direct connections between them.

This allows autonomous communication and or synchronization between peripherals, saving CPU resources thus power supply consumption.

In addition, these hardware connections remove software latency and allow design of predictable system.

Depending on peripherals, these interconnections can operate in Run, Sleep, Low-power run and sleep, Stop 0 and Stop 1 modes.

11.2 Connection summary

Table 60. STM32G4 Series peripherals interconnect matrix[1] [2]

Source	Destination																																			
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM8	TIM15	TIM16	TIM17	TIM20	LPTIM1	HRTIM	ADC1	ADC2	ADC3	ADC4	ADC5	OPAMP1	OPAMP2	OPAMP3	OPAMP4	OPAMP5	OPAMP6	DAC1	DAC2	DAC3	DAC4	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	IRTIM
TIM1	-	1	1	1	1	-	-	1	1	-	-	1	-	10 14	19	19	19	19	19	-	-	-	-	-	-	-	-	20	-	15	15	15	15	15	15	15	-
TIM2	1	-	1 2	-	1 2	-	-	1	1	-	-	1	-	10	19	19	19	19	19	-	-	-	-	-	-	20	20	20	20	15	15	15	-	15	15	-	-
TIM3	1	1 2	-	1 2	1 2	-	-	1	1	-	-	1	-	10	19	19	19	19	19	-	-	-	-	-	-	20	20	20	20	15	15	15	15	15	-	15	-
TIM4	1	1 2	1 2	-	1	-	-	1	1	-	-	1	-	-	19	19	19	19	19	-	-	-	-	-	-	20	20	20	20	-	-	-	-	-	-	-	-
TIM5	1	1 2	1	1 2	-	-	-	1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM6	-	-	-	-	-	-	-	-	-	-	-	-	-	10 13	19	19	19	19	19	-	-	-	-	-	-	20	20	20	20	-	-	-	-	-	-	-	-
TIM7	-	-	-	-	-	-	-	-	-	-	-	-	-	10 12	19	19	19	19	19	-	-	-	-	-	-	20	20	20	20	-	-	-	-	-	-	-	-
TIM8	1	1	1	1	1	-	-	-	1	-	-	1	-	-	19	19	19	19	19	-	-	-	-	-	-	20	20	-	20	15	15	15	15	15	15	15	-
TIM15	1	1	1	1	1	-	-	1	-	-	-	1	-	10	19	19	19	19	19	-	-	-	-	-	-	20	20	20	20	-	-	-	15	-	15	15	-
TIM16	1	1	1	1	1	-	-	1	1	-	-	1	-	12 13	19	19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	16
TIM17	1	1	1	1	1	-	-	1	1	-	-	1	-	12 13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	16
TIM20	1	1	1	1	1	-	-	1	1	-	-	1	-	-	19	19	19	19	19	-	-	-	-	-	-	-	-	-	-	15	15	15	15	15	15	15	-
LPTIM1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	19	19	19	19	19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HRTIM	1	1	1	1	1	-	-	1	1	-	-	1	-	-	19	19	19	19	19	-	-	-	-	-	-	20	20	20	20	-	-	-	-	-	-	-	-
ADC1	2	-	2	-	-	-	-	2	-	-	-	-	-	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ADC2	2	-	2	-	-	-	-	2	-	-	-	-	-	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ADC3	2	-	-	-	-	-	-	2	-	-	-	2	-	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ADC4	2	-	-	-	-	-	-	2	-	-	-	2	-	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ADC5	2	-	-	-	-	-	-	2	-	-	-	2	-	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
T. Sensor	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21	-	-	-	21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VBAT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21	-	21	-	21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VREFINT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21	-	21	21	21	-	-	-	-	-	-	-	-	-	-	23	23	23	23	23	23	23	-
OPAMP1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21 24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OPAMP2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21 24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OPAMP3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21 24	21 24	21 24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OPAMP4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21 24	-	-	21 24	21 24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OPAMP5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21 24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 60. STM32G4 Series peripherals interconnect matrix[(1)] [(2)] (continued)

Source	Destination																																			
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM8	TIM15	TIM16	TIM17	TIM20	LPTIM1	HRTIM	ADC1	ADC2	ADC3	ADC4	ADC5	OPAMP1	OPAMP2	OPAMP3	OPAMP4	OPAMP5	OPAMP6	DAC1	DAC2	DAC3	DAC4	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	IRTIM
OPAMP6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21 24	21 24	-	21 24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAC1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	22 23	22 23	22 23	22 23	22 23	-	-	-
DAC2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	22 23	22 23	-
DAC3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	22 24	-	22 24	-	-	22 24	-	-	-	-	22 23	22 23	22 23	22 23	-	-	-	-
DAC4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	22 24	22 24	-	-	-	-	-	-	-	-	-	22 23	22 23	22 23	-
HSE	-	-	-	-	-	-	-	-	-	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LSE	-	2	-	-	4	-	-	-	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HSI16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LSI	-	-	-	-	4	-	-	-	-	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MCO	-	-	-	-	-	-	-	-	-	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EXTI	-	-	-	-	-	-	-	-	-	-	-	-	-	-	19	19	19	19	19	-	-	-	-	-	-	20	20	20	20	-	-	-	-	-	-	-	-
RTC	-	-	-	-	4	-	-	-	-	4	4	-	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP1	2 3 4 8 9	2 3 4 5 7	2 3 4 5	2 4 5	2 4 5	-	-	2 3 4 8 9	3 4 8	3 8	3 8	2 3 4 8 9	17	10 11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP2	2 3 4 8 9	2 3 4 5 7	2 3 4 5	2 4 5	2 4 5	-	-	2 3 4 8 9	3 5 8	3 8	3 8	2 3 4 8 9	17	10 11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP3	2 3 4 8 9	2 3 4 5	2 3 4 5 6	2 4 5	2 4 5	-	-	2 3 4 8 9	3 5 8	3 8	3 8	2 3 4 8 9	17	10 11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP4	2 3 4 8 9	2 3 4 5 6	2 3 4 5	2 4 5	2 4 5	-	-	2 3 4 8 9	3 8	3 8	3 8	2 3 4 8 9	17	10 11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP5	2 3 8 9	2 3 4	2 3 4 5	2 4 5 6	2 4 5	-	-	2 3 8 9	3 4 8	3 8	3 4 8	2 3 8 9	-	10 11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 60. STM32G4 Series peripherals interconnect matrix[1] [2] (continued)

Source	Destination																																				
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM8	TIM15	TIM16	TIM17	TIM20	LPTIM1	HRTIM	ADC1	ADC2	ADC3	ADC4	ADC5	OPAMP1	OPAMP2	OPAMP3	OPAMP4	OPAMP5	OPAMP6	DAC1	DAC2	DAC3	DAC4	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	IRTIM
COMP6	2 3 8 9	2 3 5	2 3 4 5	2 4 5	2 4 5	-	-	2 3 8 9	3 5 8	3 5 8	3 8	2 3 8 9	-	10 11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP7	2 3 8 9	2 3	2 3 4 5	2 4 5	2 4 5	-	-	2 3 8 9	3 4 5 8	3 8	3 8	2 3 8 9	-	10 11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SYST ERR	18	-	-	-	-	-	-	18	18	18	18	18	-	18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. Numbers inside table link to corresponding interconnect number detailed in *Section 11.3: Interconnection details*.
2. The “-” symbol in grayed cells means no interconnect.

11.3 Interconnection details

11.3.1 From timer (TIMx, HRTIM) to timer (TIMx)

Table 61. Interconnect 1

Timer input trigger signal	Timer input trigger source assignement							
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM15	TIM20
timx_itr0	-	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
timx_itr1	tim2_trgo	-	tim2_trgo	tim2_trgo	tim2_trgo	tim2_trgo	tim2_trgo	tim2_trgo
timx_itr2	tim3_trgo	tim3_trgo	-	tim3_trgo	tim3_trgo	tim3_trgo	tim3_trgo	tim3_trgo
timx_itr3	tim4_trgo	tim4_trgo	tim4_trgo	-	tim4_trgo	tim4_trgo	tim4_trgo	tim4_trgo
timx_itr4	tim5_trgo	tim5_trgo	tim5_trgo	tim5_trgo	-	tim5_trgo	tim5_trgo	tim5_trgo
timx_itr5	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo	-	tim8_trgo	tim8_trgo
timx_itr6	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo	-	tim15_trgo
timx_itr7	tim16_oc	tim16_oc	tim16_oc	tim16_oc	tim16_oc	tim16_oc	tim16_oc	tim16_oc
timx_itr8	tim17_oc	tim17_oc	tim17_oc	tim17_oc	tim17_oc	tim17_oc	tim17_oc	tim17_oc
timx_itr9	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo	-
timx_itr10	hrtim_out_scout2	hrtim_out_scout2	hrtim_out_scout2	hrtim_out_scout2	hrtim_out_scout2	hrtim_out_scout2	hrtim_out_scout2	hrtim_out_scout2

The HRTIM burst operation can be triggered by on chip event coming from other general purpose timer.

The burst mode controller counter can be clocked by general purpose timers as well as shown in the *Table 62*.

Table 62. Interconnect 12

HRTIM Burst mode trigger event/ clock signal	HRTIM Burst mode trigger event/ clock signal assignment
hrtim_bm_trg	tim7_trgo
hrtim_bm_ck1	tim16_oc
hrtim_bm_ck2	tim17_oc
hrtim_bm_ck3	tim7_trgo‘

Table 63. Interconnect 13

HRTIM update enable signal	HRTIM update enable assignment
hrtim_upd_en1	tim16_oc
hrtim_upd_en2	tim17_oc
hrtim_upd_en3	tim6_oc

The HRTIM can be synchronized by external sources as shown in the *Table 64*.

Table 64. Interconnect 14

HRTIM synchronization signals	HRTIM synchronization signal assignment
hrtim_in_sync2	tim1_trgo
hrtim_in_sync3	HRTIM_SCIN

Some of 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.

A description of the feature is provided in: *Section 28.3.30: Timer synchronization*.

The modes of synchronization are detailed in:

- *Section 28.3.30: Timer synchronization* for advanced-control timers (TIM1/TIM8/TIM20)
- *Section 29.4.23: Timer synchronization* for general-purpose timers (TIM2/TIM3/TIM4/TIM5)
- *Section 30.4.25: Timer synchronization (TIM15)* for general-purpose timer (TIM15)

Triggering signals

The output (from Master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1/TIM8/TIM20) following a configurable timer event.

The input (to slave) is on signals TIMx_ITRx

The input and output signals for TIM1/TIM8/TIM20 are shown in *Figure 269: Advanced-control timer block diagram*.

The possible master/slave connections are given in:

- *Table 250: TIMx internal trigger connection*

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

11.3.2 From timer (TIMx, HRTIM) and EXTI to ADC (ADCx)

Table 65. Interconnect 19

ADC trigger selection EXTSEL[4:0] or JEXTSEL[4:0]	ADC triggers signals assignment			
	ADC1/2		ADC3/4/5	
	Regular	Injected	Regular	Injected
0	tim1_cc1	tim1_trgo	tim3_cc1	tim1_trgo
1	tim1_cc2	tim1_cc4	tim2_cc3	tim1_cc4
2	tim1_cc3	tim2_trgo	tim1_cc3	tim2_trgo
3	tim2_cc2	tim2_cc1	tim8_cc1	tim8_cc2
4	tim3_trgo	tim3_cc4	tim3_trgo	tim4_cc3
5	tim4_cc4	tim4_trgo	exti2	tim4_trgo
6	exti11	exti15	tim4_cc1	tim4_cc4
7	tim8_trgo	tim8_cc4	tim8_trgo	tim8_cc4
8	tim8_trgo2	tim1_trgo2	tim8_trgo2	tim1_trgo2
9	tim1_trgo	tim8_trgo	tim1_trgo	tim8_trgo
10	tim1_trgo2	tim8_trgo2	tim1_trgo2	tim8_trgo2
11	tim2_trgo	tim3_cc3	tim2_trgo	tim1_cc3
12	tim4_trgo	tim3_trgo	tim4_trgo	tim3_trgo
13	tim6_trgo	tim3_cc1	tim6_trgo	exti3
14	tim15_trgo	tim6_trgo	tim15_trgo	tim6_trgo
15	tim3_cc4	tim15_trgo	tim2_cc1	tim15_trgo
16	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo
17	tim20_trgo2	tim20_trgo2	tim20_trgo2	tim20_trgo2
18	tim20_cc1	tim20_cc4	tim20_cc1	tim20_cc2
19	tim20_cc2	hrtim_adc_trg2	hrtim_adc_trg2	hrtim_adc_trg2

Table 65. Interconnect 19 (continued)

ADC trigger selection EXTSEL[4:0] or JEXTSEL[4:0]	ADC triggers signals assignment			
	ADC1/2		ADC3/4/5	
	Regular	Injected	Regular	Injected
20	tim20_cc3	hrtim_adc_trg4	hrtim_adc_trg4	hrtim_adc_trg4
21	hrtim_adc_trg1	hrtim_adc_trg5	hrtim_adc_trg1	hrtim_adc_trg5
22	hrtim_adc_trg3	hrtim_adc_trg6	hrtim_adc_trg3	hrtim_adc_trg6
23	hrtim_adc_trg5	hrtim_adc_trg7	hrtim_adc_trg5	hrtim_adc_trg7
24	hrtim_adc_trg6	hrtim_adc_trg8	hrtim_adc_trg6	hrtim_adc_trg8
25	hrtim_adc_trg7	hrtim_adc_trg9	hrtim_adc_trg7	hrtim_adc_trg9
26	hrtim_adc_trg8	hrtim_adc_trg10	hrtim_adc_trg8	hrtim_adc_trg10
27	hrtim_adc_trg9	TIM16_CC1	hrtim_adc_trg9	hrtim_adc_trg1
28	hrtim_adc_trg10	-	hrtim_adc_trg10	hrtim_adc_trg3
29	lptim_out	lptim_out	lptim_out	lptim_out
30	tim7_trgo	tim7_trgo	tim7_trgo	tim7_trgo
31	-	-	-	-

Timers (TIMx, HRTIM) can be used to generate an ADC triggering event.

TIMx synchronization is described in: *Section 28.3.31: ADC synchronization* (TIM1/TIM8).

ADC synchronization is described in: *Section 21.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*.

Triggering signals

The output (from timer) is on signal TIMx_TRGO, TIMx_TRGO2 or TIMx_CCx event.

The input (to ADC) is on signal EXT[15:0], JEXT[15:0].

The connection between timers and ADC is provided in:

- *Table 163: ADC1/2 - External triggers for regular channels*
- *Table 164: ADC1/2 - External trigger for injected channels*

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

11.3.3 From ADC (ADCx) to timer (TIMx, HRTIM)

See please *Table 73: Interconnect 2* and *Table 81.: Interconnect 10*

ADCs Analog watchdogs are connected to TIM1/8/20 for digital power applications (cycle-bycycle current regulation with ADC).

A description of the ADC analog watchdog setting is provided in: *Section 21.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*.

Trigger settings on the timer are provided in: *Section 28.3.6: External trigger input*.

Triggering signals

The output (from ADC) is on signals ADCn_AWDx_OUT and the input (to timer) on signal TIMx_ETR (external trigger) or hrtim_eevx[4:1].

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

11.3.4 From timer (TIMx, HRTIM) and EXTI to DAC (DACx)

Table 66. Interconnect 20

DAC trigger selection (TSELx[3:0], STRSTTRIG SELx[3:0])	DAC triggers signals assignment							
	DAC1		DAC2		DAC3		DAC4	
	Update/ reset	Inc	Update/ reset	Inc	Update /reset	Inc	Update/ reset	Inc
0	sw	-	sw	-	sw	-	sw	-
1	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo	tim1_trgo	tim1_trgo	tim8_trgo	tim8_trgo
2	tim7_trgo	tim7_trgo	tim7_trgo	tim7_trgo	tim7_trgo	tim7_trgo	tim7_trgo	tim7_trgo
3	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo
4	tim2_trgo	tim2_trgo	tim2_trgo	tim2_trgo	tim2_trgo	tim2_trgo	tim2_trgo	tim2_trgo
5	tim4_trgo	tim4_trgo	tim4_trgo	tim4_trgo	tim4_trgo	tim4_trgo	tim4_trgo	tim4_trgo
6	exti9	exti10	exti9	exti10	exti9	exti10	exti9	exti10
7	tim6_trgo	tim6_trgo	tim6_trgo	tim6_trgo	tim6_trgo	tim6_trgo	tim6_trgo	tim6_trgo
8	tim3_trgo	tim3_trgo	tim3_trgo	tim3_trgo	tim3_trgo	tim3_trgo	tim3_trgo	tim3_trgo
9	hrtim_dac_reset_trg1	hrtim_step_trig1	hrtim_dac_reset_trg1	hrtim_step_trig1	hrtim_dac_reset_trg1	hrtim_step_trig1	hrtim_dac_reset_trg1	hrtim_step_trig1
10	hrtim_dac_reset_trg2	hrtim_step_trig2	hrtim_dac_reset_trg2	hrtim_step_trig2	hrtim_dac_reset_trg2	hrtim_steptrig_2	hrtim_rst__trig_2	hrtim_step_trig2
11	hrtim_dac_reset_trg3	hrtim_step_trig3	hrtim_dac_reset_trg3	hrtim_step_trig3	hrtim_dac_reset_trg3	hrtim_step_trig3	hrtim_dac_reset_trg3	hrtim_step_trig3

Table 66. Interconnect 20 (continued)

DAC trigger selection (TSELx[3:0], STRSTTRIGSELx[3:0])	DAC triggers signals assignment							
	DAC1		DAC2		DAC3		DAC4	
	Update/reset	Inc	Update/reset	Inc	Update/reset	Inc	Update/reset	Inc
12	hrtim_dac_reset_trg4	hrtim_step_trig4	hrtim_dac_reset_trg4	hrtim_step_trig4	hrtim_dac_reset_trg4	hrtim_step_trig4	hrtim_dac_reset_trg4	hrtim_step_trig4
13	hrtim_dac_reset_trg5	hrtim_step_trig5	hrtim_dac_reset_trg5	hrtim_step_trig5	hrtim_dac_reset_trg5	hrtim_step_trig5	hrtim_dac_reset_trg5	hrtim_step_trig5
14	hrtim_dac_reset_trg6	hrtim_step_trig6	hrtim_dac_reset_trg6	hrtim_step_trig6	hrtim_dac_reset_trg6	hrtim_step_trig6	hrtim_dac_reset_trg6	hrtim_step_trig6
15	hrtim_dac_trg1	-	hrtim_dac_trg2	-	hrtim_dac_trg3	-	hrtim_dac_trg1	-

Timers (TIMx, HRTIM) and EXTI can be used as triggering event to start a DAC conversion.

Triggering signals

The output (from timer) is on signal TIMx_TRGO directly connected to corresponding DAC inputs.

Selection of input triggers on DAC is provided in *Section 22.4.7: DAC trigger selection* (single and dual mode).

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

11.3.5 From HSE, LSE, LSI, HSI16, MCO, RTC to timer (TIMx)

See please *Table 75: Interconnect 4* and *Table 73: Interconnect 2*

External clocks (HSE, LSE), internal clocks (LSI, HSI16), microcontroller output clock (MCO), GPIO and RTC wakeup interrupt can be used as input to timer (TIMx).

This allows to calibrate the HSI16 and precisely measure the LSI oscillator frequency.

When Low Speed External (LSE) oscillator is used, no additional hardware connections are required.

This feature is described in *Section 7.2.16: Internal/external clock measurement with TIM5/TIM15/TIM16/TIM17*.

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

11.3.6 From RTC, COMPx to low-power timer (LPTIM1)

Table 67. Interconnect 17

LPTIM trigger input signal (TRIGSEL[3:0])	LPTIM1 trigger source assignment
lptim1_ext_trig0	LPTIM1_ETR
lptim1_ext_trig1	rtc_alra_trg
lptim1_ext_trig2	rtc_alrb_trg
lptim1_ext_trig3	RTC_TAMP1
lptim1_ext_trig4	RTC_TAMP2
lptim1_ext_trig5	RTC_TAMP3
lptim1_ext_trig6	comp1_out
lptim1_ext_trig7	comp2_out
lptim1_ext_trig8	comp3_out
lptim1_ext_trig9	comp4_out
lptim1_ext_trig10	comp5_out
lptim1_ext_trig11	comp6_out
lptim1_ext_trig12	comp7_out

RTC alarm A/B, RTC_TAMP1/2/3 input detection, COMPx_OUT can be used as trigger to start LPTIM counters (LPTIM1).

Triggering signals

This trigger feature is described in *Section 32.4.6: Trigger multiplexer* (and following sections).

The input selection is described in *Table 310: LPTIM1 external trigger connection*.

Active power mode

Run, Sleep, Low-power run, Low-power sleep, Stop 0, Stop 1.

11.3.7 From timer (TIMx) to comparators (COMPx)

Table 68. Interconnect 15

Comparator blanking signal BLANKSEL[2:0]	Comparator blanking source assignment						
	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7
1	tim1_oc5	tim1_oc5	tim1_oc5	tim3_oc4	tim2_oc3	tim8_oc5	tim1_oc5
2	tim2_oc3	tim2_oc3	tim3_oc3	tim8_oc5	tim8_oc5	tim2_oc4	tim8_oc5
3	tim3_oc3	tim3_oc3	tim2_oc4	tim15_oc1	tim3_oc3	tim15_oc2	tim3_oc3
4	tim8_oc5	tim8_oc5	tim8_oc5	tim1_oc5	tim1_oc5	tim1_oc5	tim15_oc2
5	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5
6	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1
7	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3

Timers (TIMx) can be used as blanking window to COMPx (x = 1...7)

The blanking function is described in *Section 24.3.6: COMP output blanking*.

The blanking sources are given in comparators control and status register (COMP_CxCSR) bits BLANKSEL[2:0].

Triggering signals

Timer output signal TIMx_Ocx are the inputs to blanking source of COMPx.

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

11.3.8 From internal analog source to ADC (ADCx), comparator (COMPx) and OPAMP (OPAMPx)

Table 69 provides the ADC channels mapping on GPIOs or internal connections to temperature sensor (VTS), internal reference voltage VREFINT, VBAT/3 or opampxint_vout.

Table 69. Interconnect 21

ADC channel number	ADC channel source assignment				
	ADC1	ADC2	ADC3	ADC4	ADC5
IN0	-	-	-	-	-
IN1	PA0	PA0	PB1/OPAMP3 _VOUT	PE14	PA8/OPAMP5 _VOUT
IN2	PA1	PA1	PE9	PE15	PA9
IN3	PA2/OPAMP1 _VOUT	PA6/OPAMP2 _VOUT	PE13	PB12/OPAMP4 _VOUT	opamp5_int _vout(1)
IN4	PA3	PA7	PE7	PB14	Temp sensor

Table 69. Interconnect 21 (continued)

ADC channel number	ADC channel source assignment				
	ADC1	ADC2	ADC3	ADC4	ADC5
IN5	PB14	PC4	PB13	PB15	opamp4_int _vout[1]
IN6	PC0	PC0	PE8	PE8	PE8
IN7	PC1	PC1	PD10	PD10	PD10
IN8	PC2	PC2	PD11	PD11	PD11
IN9	PC3	PC3	PD12	PD12	PD12
IN10	PF0	PF1	PD13	PD13	PD13
IN11	PB12/OPAMP4 _VOUT	PC5	PD14	PD14	PD14
IN12	PB1/OPAMP3 _VOUT	PB2	PB0	PD8	PD8
IN13	opamp1_int _vout[1]	PA5	opamp3_int _vout[1]	PD9	PD9
IN14	PB11/OPAMP6 _VOUT	PB11/OPAMP6 _VOUT	PE10	PE10	PE10
IN15	PB0	PB15	PE11	PE11	PE11
IN16	Temp sensor	opamp2_int _vout[1]	PE12	PE12	PE12
IN17	$V_{BAT}/3$	PA4	$V_{BAT}/3$[2] opamp6_int_ vout[1][3]	opamp6_int _vout[1]	$V_{BAT}/3$
IN18	V_{REFINT}	opamp3_int _vout[1]	V_{REFINT}	V_{REFINT}	V_{REFINT}

1. Internal OPAMP output connected directly to ADC input only (no available externally on a pin).
2. For Category 3 devices only.
3. For Category 4 devices only.

The DAC outputs are available on GPIOs or can be internally connected to comparators and operational amplifiers.

Table 70. Interconnect 22

Destination	Source						
	DAC1		DAC2	DAC3		DAC4	
	CH1	CH2	CH1	CH1	CH2	CH1	CH2
GPIO	PA4	PA5	PA6	-	-	-	-
COMP1	x	-	-	x	-	-	-
COMP2	-	x	-	-	x	-	-

Table 70. Interconnect 22 (continued)

Destination	Source						
	DAC1		DAC2	DAC3		DAC4	
	CH1	CH2	CH1	CH1	CH2	CH1	CH2
COMP3	x	-	-	x	-	-	-
COMP4	x	-	-	-	x	-	-
COMP5	-	x	-	-	-	x	-
COMP6	-	-	x	-	-		x
COMP7	-	-	x	-	-	x	-
OPAMP1	-	-	-	x	-	-	-
OPAMP2	-	-	-	-	-	-	-
OPAMP3	-	-	-	-	x	-	-
OPAMP4	-	-	-	-	-	x	-
OPAMP5	-	-	-	-	-	-	x
OPAMP6	-	-	-	x	-	-	-

Table 71. Interconnect 23

Comparator input/output signals		Comparators input/output signals assignment										
		GPIO					DAC output[1]		VREFINT scaler			
COMP1	INP	PA1	PB1	-	-	-	-	-	-	-	-	-
	INM	PA0	PA4	-	-	-	DAC1_CH1 PA4	DAC3_CH1	VREFINT	3/4 VREFINT	1/2 VREFINT	1/4 VREFINT
	OUT	PA0	PF4	PA6	PA11	PB8	-	-	-	-	-	-
COMP2	INP	PA3	PA7	-	-	-	-	-	-	-	-	-
	INM	PA2	PA5	-	-	-	DAC1_CH2 PA5	DAC3_CH2	VREFINT	3/4 VREFINT	1/2 VREFINT	1/4 VREFINT
	OUT	PA2	PA7	PA12	PB9	-	-	-	-	-	-	-
COMP3	INP	PC1	PA0	-	-	-	-	-	-	-	-	-
	INM	PC0	PF1	-	-	-	DAC1_CH1 PA4	DAC3_CH1	VREFINT	3/4 VREFINT	1/2 VREFINT	1/4 VREFINT
	OUT	PC2	PB7	PB15	-	-	-	-	-	-	-	-

Table 71. Interconnect 23 (continued)

Comparator input/output signals		Comparators input/output signals assignment										
		GPIO					DAC output[1]		VREFINT scaler			
COMP4	INP	PB0	PE7	-	-	-	-	-	-	-	-	-
	INM	PB2	PE8	-	-	-	DAC1_CH1 PA4	DAC3_CH2	VREF INT	3/4 VREF INT	1/2 VREF INT	1/4 VREF INT
	OUT	PB1	PB6	PB14	-	-	-	-	-	-	-	-
COMP5	INP	PD12	PB13	-	-	-	-	-	-	-	-	-
	INM	PD13	PB10	-	-	-	DAC1_CH2 PA5	DAC4_CH1	VREF INT	3/4 VREF INT	1/2 VREF INT	1/4 VREF INT
	OUT	PC7	PA9	-	-	-	-	-	-	-	-	-
COMP6	INP	PD11	PB11	-	-	-	-	-	-	-	-	-
	INM	PD10	PB15	-	-	-	DAC2_CH1 PA6	DAC4_CH2	VREF INT	3/4 VREF INT	1/2 VREF INT	1/4 VREF INT
	OUT	PC6	PA10	-	-	-	-	-	-	-	-	-
COMP7	INP	PD14	PB14	-	-	-	-	-	-	-	-	-
	INM	PD15	PB12	-	-	-	DAC2_CH1 PA6	DAC4_CH1	VREF INT	3/4 VREF INT	1/2 VREF INT	1/4 VREF INT
	OUT	PC8	PA8	-	-	-	-	-	-	-	-	-

1. It is an internal connection.

Table 72. Interconnect 24

Operational amplifier input/output signals		Operational amplifier input/output signals assignment						
		GPIO				DAC output	ADC input on GPIO	ADC internal input
OPAMP1	VINP	PA1	PA3	PA7	-	DAC3_CH1	-	-
	VINM	PA3	PC5	-	-	-	-	-
	VOUT	PA2		-	-	-	ADC1_IN3	ADC1_IN13
OPAMP2	VINP	PA7	PB14	PB0	PD14	-	-	-
	VINM	PA5	PC5	-	-	-	-	-
	VOUT	PA6		-	-	-	ADC2_IN3	ADC2_IN16
OPAMP3	VINP	PB0	PB13	PA1	-	DAC3_CH2	-	-
	VINM	PB2	PB10	-	-	-	-	-
	VOUT	PB1	-	-	-	-	ADC3_IN1/ ADC1_IN12	ADC2_IN18/ ADC3_IN13

Table 72. Interconnect 24 (continued)

Operational amplifier input/output signals		Operational amplifier input/output signals assignment						
		GPIO				DAC output	ADC input on GPIO	ADC internal input
OPAMP4	VINP	PB13	PD11	PB11	-	DAC4_CH1	-	-
	VINM	PB10	PD8	-	-	-	-	-
	VOUT	PB12	-	-	-	-	ADC4_IN3/ ADC1_IN11	ADC5_IN5
OPAMP5	VINP	PB14	PD12	PC3	-	DAC4_CH2	-	-
	VINM	PB15	PA3	-	-	-	-	-
	VOUT	PA8		-	-	-	ADC5_IN1	ADC5_IN3
OPAMP6	VINP	PB12	PD9	PB13	-	DAC3_CH1	-	-
	VINM	PA1	PB1	-	-	-	-	-
	VOUT	PB11	-	-	-	-	ADC12_IN14	ADC4_IN17

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

11.3.9 From comparators (COMPx) to timers (TIMx, HRTIM)

Comparators (COMPx) output values can be connected to:

- Timers (TIMx) input captures or TIMx_ETR signals or TIMx_OCREFCLR signals.
- hrtim_eevx[4:1] and hrtim_in_fltx[4:1]

Comparators (COMPx) output values can also generate break input signals for timers (TIMx) see Section 30.3.17: Bidirectional break inputs.

Table 73. Interconnect 2

Timer external trigger input signal	Timer external trigger signals assignment						
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM20
timx_etr0	TIM1_ETR	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR	TIM8_ETR	TIM20_ETR
timx_etr1	comp1_out	comp1_out	comp1_out	comp1_out	comp1_out	comp1_out	comp1_out
timx_etr2	comp2_out	comp2_out	comp2_out	comp2_out	comp2_out	comp2_out	comp2_out
timx_etr3	comp3_out	comp3_out	comp3_out	comp3_out	comp3_out	comp3_out	comp3_out
timx_etr4	comp4_out	comp4_out	comp4_out	comp4_out	comp4_out	comp4_out	comp4_out
timx_etr5	comp5_out	comp5_out	comp5_out	comp5_out	comp5_out	comp5_out	comp5_out

Table 73. Interconnect 2 (continued)

Timer external trigger input signal	Timer external trigger signals assignment						
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM20
timx_etr6	comp6_out	comp6_out	comp6_out	comp6_out	comp6_out	comp6_out	comp6_out
timx_etr7	comp7_out	comp7_out	comp7_out	comp7_out	comp7_out	comp7_out	comp7_out
timx_etr8	adc1_awd1	tim3_etr	tim2_etr	tim3_etr	tim2_etr	adc2_awd1	adc3_awd1
timx_etr9	adc1_awd2	tim4_etr	tim4_etr	tim5_etr	tim3_etr	adc2_awd2	adc3_awd2
timx_etr10	adc1_awd3	tim5_etr	-	-	-	adc2_awd3	adc3_awd3
timx_etr11	adc4_awd1	lse_css_out	adc2_awd1	-	-	adc3_awd1	adc5_awd1
timx_etr12	adc4_awd2	-	adc2_awd2	-	-	adc3_awd2	adc5_awd2
timx_etr13	adc4_awd3	-	adc2_awd3	-	-	adc3_awd3	adc5_awd3

Table 74. Interconnect 3

Timer OCREF clear signal	Timer OCREF clear signals assignment							
	TIM1	TIM2	TIM3	TIM8	TIM15	TIM16	TIM17	TIM20
timx_ocref_clr0	comp1_out	comp1_out	comp1_out	comp1_out	comp1_out	comp1_out	comp1_out	comp1_out
timx_ocref_clr1	comp2_out	comp2_out	comp2_out	comp2_out	comp2_out	comp2_out	comp2_out	comp2_out
timx_ocref_lr2	comp3_out	comp3_out	comp3_out	comp3_out	comp3_out	comp3_out	comp3_out	comp3_out
timx_ocref_clr3	comp4_out	comp4_out	comp4_out	comp4_out	comp4_out	comp4_out	comp4_out	comp4_out
timx_ocref_clr4	comp5_out	comp5_out	comp5_out	comp5_out	comp5_out	comp5_out	comp5_out	comp5_out
timx_ocref_clr5	comp6_out	comp6_out	comp6_out	comp6_out	comp6_out	comp6_out	comp6_out	comp6_out
timx_ocref_clr6	comp7_out	comp7_out	comp7_out	comp7_out	comp7_out	comp7_out	comp7_out	comp7_out

Table 75. Interconnect 4

Timer TI1 input signal	Timer TI1 signals assignment									
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM15	TIM16	TIM17	TIM20
timx_ ti1_in0	TIM1 external TI1 input pins	TIM2 external TI1 input pins	TIM3 external TI1 input pins	TIM4 external TI1 input pins	TIM5 external TI1 input pins	TIM8 external TI1 input pins	TIM15 external TI1 input pins	TIM16 external TI1 input pins	TIM17 external TI1 input pins	TIM20 external TI1 input pins
timx_ ti1_in1	comp1_ out	comp1_ out	comp1_ out	comp1_ out	LSI	comp1_ out	lse_ css_out	comp6_ out	comp5_ out	comp1_ out
timx_ ti1_in2	comp2_ out	comp2_ out	comp2_ out	comp2_ out	lse_ css_out	comp2_ out	comp1_ out	MCO	MCO	comp2_ out
timx_ ti1_in3	comp3_ out	comp3_ out	comp3_ out	comp3_ out	rtc_ Wakeup	comp3_ out	comp2_ out	HSE_Di v32	HSE_Di v32	comp3_ out
timx_ ti1_in4	comp4_ out	comp4_ out	comp4_ out	comp4_ out	comp1_ out	comp4_ out	comp5_ out	rtc_ Wakeup	rtc_ Wakeup	comp4_ out
timx_ ti1_in5	-	comp5_ out	comp5_ out	comp5_ out	comp2_ out	-	comp7_ out	lse_ css_out	lse_ css_out	-
timx_ ti1_in6	-	-	comp6_ out	comp6_ out	comp3_ out	-	-	LSI	LSI	-
timx_ ti1_in7	-	-	comp7_ out	comp7_ out	comp4_ out	-	-	-	-	-
timx_ ti1_in8	-	-	-	-	comp5_ out	-	-	-	-	-
timx_ ti1_in9	-	-	-	-	comp6_ out	-	-	-	-	-
timx_ ti1_in10	-	-	-	-	comp7_ out	-	-	-	-	-

Table 76. Interconnect 5

Timer TI2 input signal	Timer TI2 signals assignment							
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM15	TIM20
timx_ ti2_in0	TIM1 external TI2 input pins	TIM2 external TI2 input pins	TIM3 external TI2 input pins	TIM4 external TI2 input pins	TIM5 external TI2 input pins	TIM8 external TI2 input pins	TIM15 external TI2 input pins	TIM20 external TI2 input pins
timx_ ti2_in1	-	comp1_out	comp1_out	comp1_out	comp1_out	-	comp2_out	-
timx_ ti2_in2	-	comp2_out	comp2_out	comp2_out	comp2_out	-	comp3_out	-
timx_ ti2_in3	-	comp3_out	comp3_out	comp3_out	comp3_out	-	comp6_out	-
timx_ ti2_in4	-	comp4_out	comp4_out	comp4_out	comp4_out	-	comp7_out	-
timx_ ti2_in5	-	comp6_out	comp5_out	comp5_out	comp5_out	-	-	-
timx_ ti2_in6	-	-	comp6_out	comp6_out	comp6_out	-	-	-
timx_ ti2_in7	-	-	comp7_out	comp7_out	comp7_out	-	-	-

Table 77. Interconnect 6

Timer TI3 input signal	Timer TI3 signals assignment						
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM20
timx_ ti3_in0	TIM1 external TI3 input pins	TIM2 external TI3 input pins	TIM3 external TI3 input pins	TIM4 external TI3 input pins	TIM5 external TI3 input pins	TIM8 external TI3 input pins	TIM20 external TI3 input pins
timx_ ti3_in1	-	comp4_out	comp3_out	comp5_out	-	-	-

Table 78. Interconnect 7

Timer TI4 input signal	Timer TI4 signals assignment						
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM20
timx_ ti4_in0	TIM1 external TI4 input pins	TIM2 external TI4 input pins	TIM3 external TI4 input pins	TIM4 external TI4 input pins	TIM5 external TI4 input pins	TIM8 external TI4 input pins	TIM20 external TI4 input pins
timx_ ti4_in1	-	comp1_out	-	comp6_out	-	-	-
timx_ ti4_in2	-	comp2_out	-	-	-	-	-

The timer break input features two channels:

- A break channel which gathers both application fault (from input pins and built-in comparators) and system-level fault (clock failure, parity error, ...).
- A break2 channel which only includes application faults (from input pins and built-in comparators).

Refer to *Table 79*, *Table 80* and *Table 83*.

Table 79. Interconnect 8

	Timer break signals assignment					
	TIM1 break (tim1_bk)	**TIM8 break (tim8_bk)**	**TIM15 break (tim15_bk)**	**TIM16 break (tim16_bk)**	**TIM17 break (tim17_bk)**	**TIM20 break (tim20_bk)**
Timer break signal sources	TIM1_BKIN pin	TIM8_BKIN pin	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin	TIM20_BKIN pin
	comp1_out	comp1_out	comp1_out	comp1_out	comp1_out	comp1_out
	comp2_out	comp2_out	comp2_out	comp2_out	comp2_out	comp2_out
	comp3_out	comp3_out	comp3_out	comp3_out	comp3_out	comp3_out
	comp4_out	comp4_out	comp4_out	comp4_out	comp4_out	comp4_out
	comp5_out	comp5_out	comp5_out	comp5_out	comp5_out	comp5_out
	comp6_out	comp6_out	comp6_out	comp6_out	comp6_out	comp6_out
	comp7_out	comp7_out	comp7_out	comp7_out	comp7_out	comp7_out

Table 80. Interconnect 9

	Timer break2 signals assignment		
	TIM1 break2 (tim1_bk2)	**TIM8 break2 (tim8_bk2)**	**TIM20 break2 (tim20_bk2)**
Timer break2 signal source	TIM1_BKIN2 pin	TIM8_BKIN2 pin	TIM20_BKIN2 pin
	comp1_out	comp1_out	comp1_out
	comp2_out	comp2_out	comp2_out
	comp3_out	comp3_out	comp3_out
	comp4_out	comp4_out	comp4_out
	comp5_out	comp5_out	comp5_out
	comp6_out	comp6_out	comp6_out
	comp7_out	comp7_out	comp7_out

Table 81. Interconnect 10

HRTIM external event input signal	HRTIM external event signal assignment			
	EExSRC[1:0]=0 (from GPIO pin)	EExSRC[1:0]=1	EExSRC[1:0]=2	EExSRC[1:0]=3
hrtim_eev1[4:1]	HRTIM_EEV1	comp2_out	tim1_trgo	adc1_AWD1
hrtim_eev2[4:1]	HRTIM_EEV2	comp4_out	tim2_trgo	adc1_AWD2
hrtim_eev3[4:1]	HRTIM_EEV3	comp6_out	tim3_trgo	adc1_AWD3
hrtim_eev4[4:1]	HRTIM_EEV4	comp1_out	comp5_out	adc2_AWD1
hrtim_eev5[4:1]	HRTIM_EEV5	comp3_out	comp7_out	adc2_AWD2
hrtim_eev6[4:1]	HRTIM_EEV6	comp2_out	comp1_out	adc2_AWD3
hrtim_eev7[4:1]	HRTIM_EEV7	comp4_out	tim7_trgo	adc3_AWD1
hrtim_eev8[4:1]	HRTIM_EEV8	comp6_out	comp3_out	adc4_AWD1
hrtim_eev9[4:1]	HRTIM_EEV9	comp5_out	tim15_trgo	comp4_out
hrtim_eev10[4:1]	HRTIM_EEV10	comp7_out	tim6_trgo	adc5_AWD1

Table 82. Interconnect 11

Fault channel	External Input FLTxSRC[1:0] = 00	On-chip source FLTxSRC[1:0] = 01	External Input FLTxSRC[1:0] = 10	On-chip source FLTxSRC[1:0] = 11
hrtim_flt1[4:1]	HRTIM_FLT1	comp2_out	EEV1_muxout	N/A
hrtim_flt2[4:1]	HRTIM_FLT2	comp4_out	EEV2_muxout	N/A
hrtim_flt3[4:1]	HRTIM_FLT3	comp6_out	EEV3_muxout	N/A
hrtim_flt4[4:1]	HRTIM_FLT4	comp1_out	EEV4_muxout	N/A
hrtim_flt5[4:1]	HRTIM_FLT5	comp3_out	EEV5_muxout	N/A
hrtim_flt6[4:1]	HRTIM_FLT6	comp5_out	EEV6_muxout	N/A

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

11.3.10 From system errors to timers (TIMx) and HRTIM

TIMx (TIM1/TIM8/TIM20/TIM15/TIM16/TIM17) break inputs and HRTIM system fault input gather MCU internal fault events coming from:

- the clock failure event generated by the clock security system (CSS),
- the PVD output,
- the SRAM parity error signal,
- the Cortex-M4 LOCKUP (Hardfault) output
- Flash ECC double error detection

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

The functionality is described in:

- *Section 28.3.18: Using the break function* (TIM1/TIM8/TIM20)
- *Section 30.4.15: Using the break function* (TIM15/TIM16/TIM17/TIM20)

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

Table 83. Interconnect 18

System error signal surce	System error signals to timer signals assignment						
	TIM1	TIM8	TIM15	TIM16	TIM17	TIM20	HRTIM
Flash ECC error	tim1_bk	tim8_bk	tim15_bk	tim16_bk	tim17_bk	tim20_bk	hrtim_sys_flt
PVD output							
SRAM1/ CCM SRAM parity							
Cortex®-M4 Lockup (hardfault)							
Clock security system (CSS)							

11.3.11 From timers (TIM16/TIM17) to IRTIM

Table 84. Interconnect 16

IRTIM control signal	IRTIM control signals assignment
modulation envelope signal	tim16_oc1
carrier signal	tim17_oc1

General-purpose timer (TIM16/TIM17) output channel TIMx_OC1 are used to generate the waveform of infrared signal output.

The functionality is described in *Section 33: Infrared interface (IRTIM)*.

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

12 Direct memory access controller (DMA)

12.1 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 (See *Table 85: DMA1 and DMA2 implementation* for number of supported channels).

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.

12.2 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^{16} - 1
- Generation of a single interrupt request for the DMA1 controller, OR-ing any of the all interrupt requests at channel level. An interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.
- Generation of an interrupt request per channel for the DMA2 controller. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.

12.3 DMA implementation

12.3.1 DMA1 and DMA2

DMA1 and DMA2 are implemented with the hardware configuration parameters shown in the table below.

Table 85. DMA1 and DMA2 implementation

Feature		DMA1	DMA2
Number of channels	Category 2 devices[1] Category 3 devices[1] Category 4 devices[1]	6 8 8	6 8 8

[1] See *Table 1: STM32G4 Series memory density*.

12.3.2 DMA request mapping

The DMA controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX peripheral.

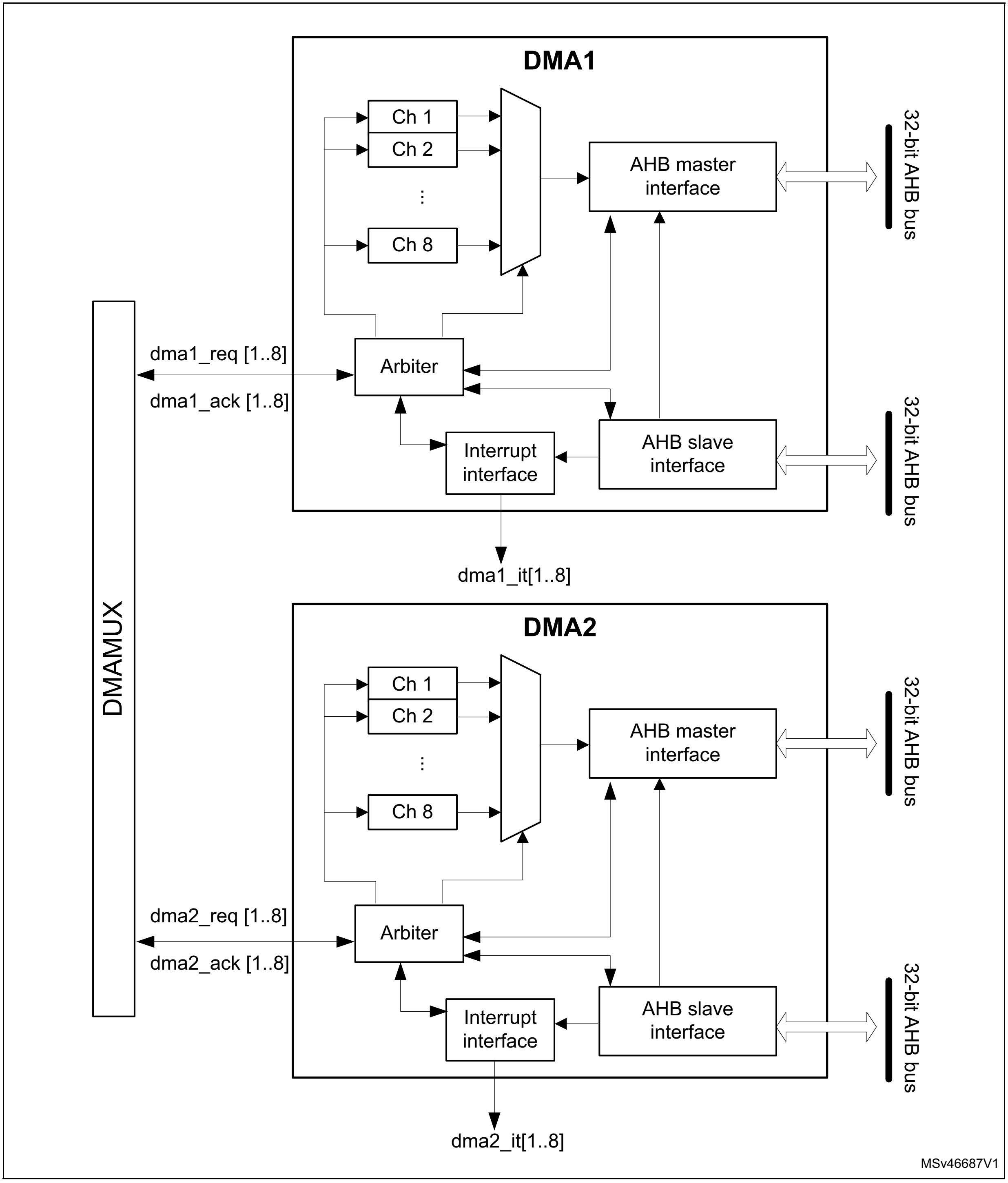
For the mapping of the different requests, refer to the *Section 13.3: DMAMUX implementation*.

12.4 DMA functional description

12.4.1 DMA block diagram

The DMA block diagram is shown in the figure below.

Figure 31. DMA block diagram



Note: See Figure 31: DMA block diagram for feature implementation.

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 an interrupt per channel to the interrupt controller.

12.4.2 DMA pins and internal signals

Table 86. 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

12.4.3 DMA transfers

The software configures the DMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB 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.

12.4.4 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.

12.4.5 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.

Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

1. Set the peripheral register address in the DMA_CPARx register.
   The data is moved from/to this address to/from the memory after the peripheral event, or after the channel is enabled in memory-to-memory mode.
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.

12.4.6 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 the table below.

Table 87. 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	4	@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.

12.4.7 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.

12.5 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 88. 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	-

12.6 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).

Note: See Figure 31: DMA block diagram for feature implementation.

12.6.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEIF8	HTIF8	TCIF8	GIF8	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
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
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

Bit 31 **TEIF8**: transfer error (TE) flag for channel 8
0: no TE event
1: a TE event occurred

Bit 30 **HTIF8**: half transfer (HT) flag for channel 8
0: no HT event
1: a HT event occurred

Bit 29 **TCIF8**: transfer complete (TC) flag for channel 8
0: no TC event
1: a TC event occurred

Bit 28 **GIF8**: global interrupt flag for channel 8
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

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

12.6.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

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
CTEIF8	CHTIF8	CTCIF8	CGIF8	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
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**
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

Bit 31 **CTEIF8**: transfer error flag clear for channel 8

Bit 30 **CHTIF8**: half transfer flag clear for channel 8

Bit 29 **CTCIF8**: transfer complete flag clear for channel 8

Bit 28 **CGIF8**: global interrupt flag clear for channel 8

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

12.6.3 DMA channel x configuration register (DMA_CCRx)

Address offset: 0x08 + 0x14 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

The register fields/bits 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.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	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 31:15 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.
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.
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.
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.
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.
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.
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.
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 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.
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.
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.
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.
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.

12.6.4 DMA channel x number of data to transfer register (DMA_CNDTRx)

Address offset: 0x0C + 0x14 * (x - 1), (x = 1 to 8)

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**
NDT[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 **NDT[15:0]**: number of data to transfer (0 to 2^{16} - 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.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

12.6.5 DMA channel x peripheral address register (DMA_CPARx)

Address offset: 0x10 + 0x14 * (x - 1), (x = 1 to 8)

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 will be 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.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

12.6.6 DMA channel x memory address register (DMA_CMARx)

Address offset: 0x14 + 0x14 * (x - 1), (x = 1 to 8)

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 will be 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.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

12.6.7 DMA register map

The table below gives the DMA register map and reset values.

Table 89. 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	TEIF8	HTIF8	TCIF8	GIF8	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	0	0	0	0

Table 89. 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	CTEIF8	CHTIF8	CTCIF8	CGIF8	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	0	0	0	0
0x008	DMA_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x00C	DMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	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.	Res.	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
0x020	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	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.	Res.	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
0x034	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	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.	Res.	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
0x048	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	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 89. 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.	Res.	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.	Res.	Res.	NDTR[15:0]															
	Reset value																	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.	Res.	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.	Res.	Res.	NDTR[15:0]															
	Reset value																	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.	Res.	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.	Res.	Res.	NDTR[15:0]															
	Reset value																	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
0x090	Reserved	Reserved.																															
0x094	DMA_CCR8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x098	DMA_CNDTR8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C	DMA_CPAR8	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
0x0A0	DMA_CMAR8	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 2.2* for the register boundary addresses.

13 DMA request multiplexer (DMAMUX)

13.1 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 13.3.1*.

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 13.3.2*.

13.2 DMAMUX main features

- Up to 16-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:
  - 115 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

13.3 DMAMUX implementation

13.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

Table 90. DMAMUX instantiation

Feature		DMAMUX
Number of DMAMUX output request channels	Category 2 devices[1] Category 3 devices[1] Category 4 devices[1]	12 16 16
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		115

[1] See *Table 1: STM32G4 Series memory density*

13.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

DMAMUX is used with DMA1 and DMA2:

For category 3 and category 4 devices:

- DMAMUX channels 0 to 7 are connected to DMA1 channels 1 to 8
- DMAMUX channels 8 to 15 are connected to DMA2 channels 1 to 8

For category 2 devices:

- DMAMUX channels 0 to 5 are connected to DMA1 channels 1 to 6
- DMAMUX channels 6 to 11 are connected to DMA2 channels 1 to 6

Table 91. DMAMUX: assignment of multiplexer inputs to resources(1)

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	DMAMUX_Req G0	44	TIM1_CH3	87	TIM20_CH2
2	DMAMUX_Req G1	45	TIM1_CH4	88	TIM20_CH3
3	DMAMUX_Req G2	46	TIM1_UP	89	TIM20_CH4
4	DMAMUX_Req G3	47	TIM1_TRIG	90	TIM20_UP
5	ADC1	48	TIM1_COM	91	AES_IN
6	DAC1_CH1	49	TIM8_CH1	92	AES_OUT
7	DAC1_CH2	50	TIM8_CH2	93	TIM20_TRIG
8	TIM6_UP	51	TIM8_CH3	94	TIM20_COM
9	TIM7_UP	52	TIM8_CH4	95	HRTIM_MASTER (hrtim_dma1)
10	SPI1_RX	53	TIM8_UP	96	HRTIM_TIMA (hrtim_dma2)
11	SPI1_TX	54	TIM8_TRIG	97	HRTIM_TIMB (hrtim_dma3)
12	SPI2_RX	55	TIM8_COM	98	HRTIM_TIMC (hrtim_dma4)
13	SPI2_TX	56	TIM2_CH1	99	HRTIM_TIMD (hrtim_dma5)
14	SPI3_RX	57	TIM2_CH2	100	HRTIM_TIME (hrtim_dma6
15	SPI3_TX	58	TIM2_CH3	101	HRTIM_TIMF (hrtim_dma7)
16	I2C1_RX	59	TIM2_CH4	102	DAC3_CH1
17	I2C1_TX	60	TIM2_UP	103	DAC3_CH2
18	I2C2_RX	61	TIM3_CH1	104	DAC4_CH1
19	I2C2_TX	62	TIM3_CH2	105	DAC4_CH2
20	I2C3_RX	63	TIM3_CH3	106	SPI4_RX
21	I2C3_TX	64	TIM3_CH4	107	SPI4_TX
22	I2C4_RX	65	TIM3_UP	108	SAI1_A
23	I2C4_TX	66	TIM3_TRIG	109	SAI1_B
24	USART1_RX	67	TIM4_CH1	110	FMAC_Read
25	USART1_TX	68	TIM4_CH2	111	FMAC_Write
26	USART2_RX	69	TIM4_CH3	112	Cordic_Read
27	USART2_TX	70	TIM4_CH4	113	Cordic_Write
28	USART3_RX	71	TIM4_UP	114	UCPD1_RX
29	USART3_TX	72	TIM5_CH1	115	UCPD1_TX
30	UART4_RX	73	TIM5_CH2	116	Reserved
31	UART4_TX	74	TIM5_CH3	117	Reserved
32	UART5_RX	75	TIM5_CH4	118	Reserved

Table 91. DMAMUX: assignment of multiplexer inputs to resources[1] (continued)

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
33	UART5_TX	76	TIM5_UP	119	Reserved
34	LPUART1_RX	77	TIM5_TRIG	120	Reserved
35	LPUART1_TX	78	TIM15_CH1	121	Reserved
36	ADC2	79	TIM15_UP	122	Reserved
37	ADC3	80	TIM15_TRIG	123	Reserved
38	ADC4	81	TIM15_COM	124	Reserved
39	ADC5	82	TIM16_CH1	125	Reserved
40	QUADSPI	83	TIM16_UP	126	Reserved
41	DAC2_CH1	84	TIM17_CH1	127	Reserved
42	TIM1_CH1	85	TIM17_UP	-	-
43	TIM1_CH2	86	TIM20_CH1	-	-

1. See *Table 2: Product specific features* for available resources.

Table 92. DMAMUX: assignment of trigger inputs to resources

Trigger input	Resource	Trigger input	Resource
0	EXTI LINE0	16	DMAMUX1_ch0_event
1	EXTI LINE1	17	DMAMUX1_ch1_event
2	EXTI LINE2	18	DMAMUX1_ch2_event
3	EXTI LINE3	19	DMAMUX1_ch3_event
4	EXTI LINE4	20	LPTIM1_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

Table 93. DMAMUX: assignment of synchronization inputs to resources

Sync. input	Resource	Sync. input	Resource
0	EXTI LINE0	16	DMAMUX1_ch0_event
1	EXTI LINE1	17	DMAMUX1_ch1_event

Table 93. DMAMUX: assignment of synchronization inputs to resources (continued)

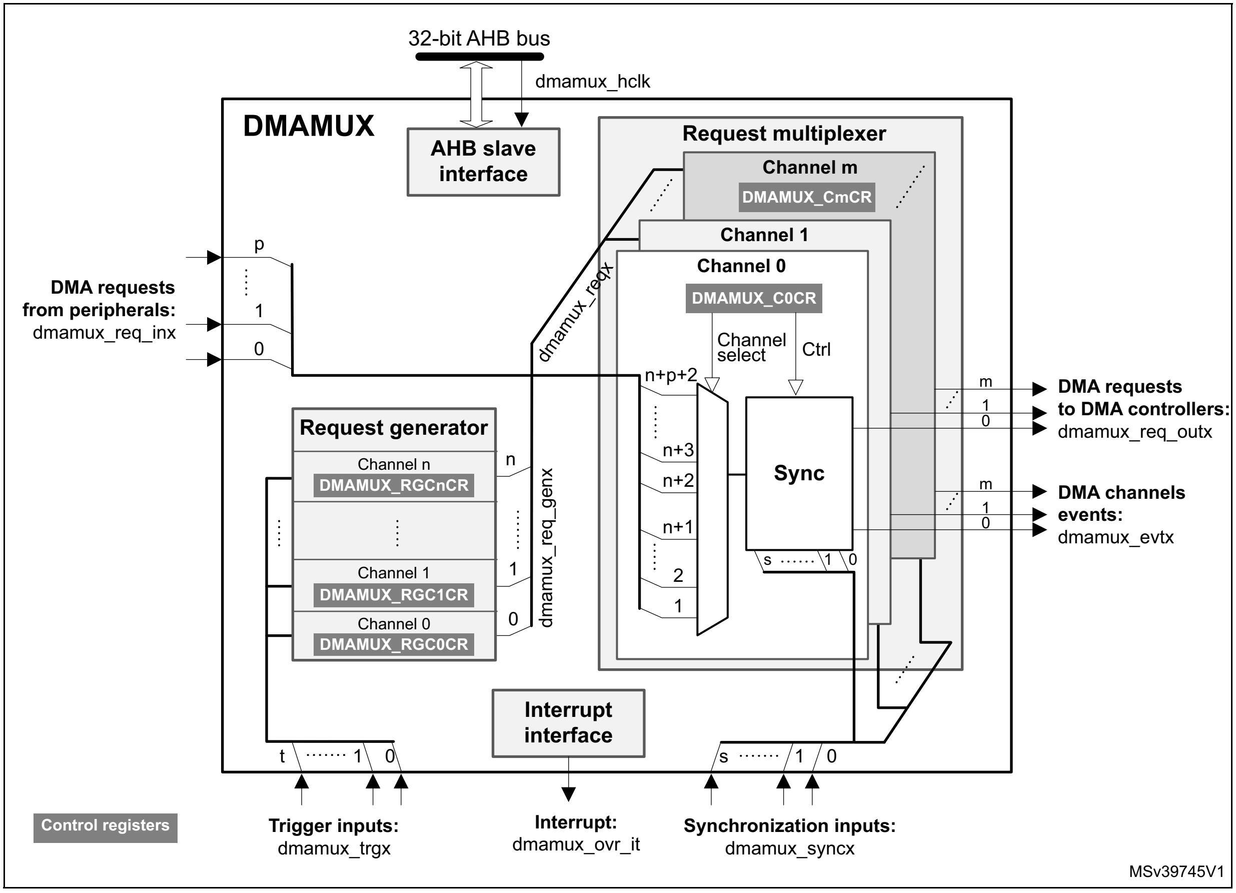
Sync. input	Resource	Sync. input	Resource
2	EXTI LINE2	18	DMAMUX1_ch2_event
3	EXTI LINE3	19	DMAMUX1_ch3_event
4	EXTI LINE4	20	LPTIM1_OUT
5	EXTI LINE5	21	Reserved
6	EXTI LINE6	22	Reserved
7	EXTI LINE7	23	Reserved
8	EXTI LINE8	24	Reserved
9	EXTI LINE9	25	Reserved
10	EXTI LINE10	26	Reserved
11	EXTI LINE11	27	Reserved
12	EXTI LINE12	28	Reserved
13	EXTI LINE13	29	Reserved
14	EXTI LINE14	30	Reserved
15	EXTI LINE15	31	Reserved

13.4 DMAMUX functional description

13.4.1 DMAMUX block diagram

Figure 32 shows the DMAMUX block diagram.

Figure 32. 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)

13.4.2 DMAMUX signals

Table 94 lists the DMAMUX signals.

Table 94. 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)
dmamux_evtx	DMAMUX events outputs
dmamux_ovr_it	DMAMUX overrun interrupts

13.4.3 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 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.

13.4.4 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 34*, 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 33. Synchronization mode of the DMAMUX request line multiplexer channel

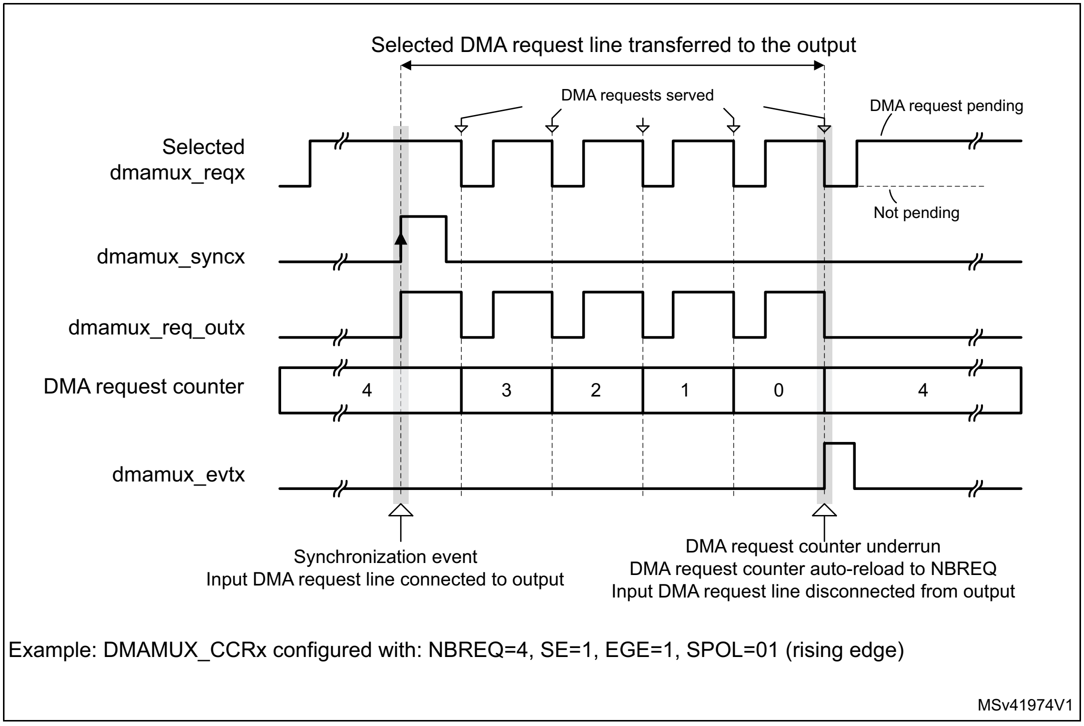
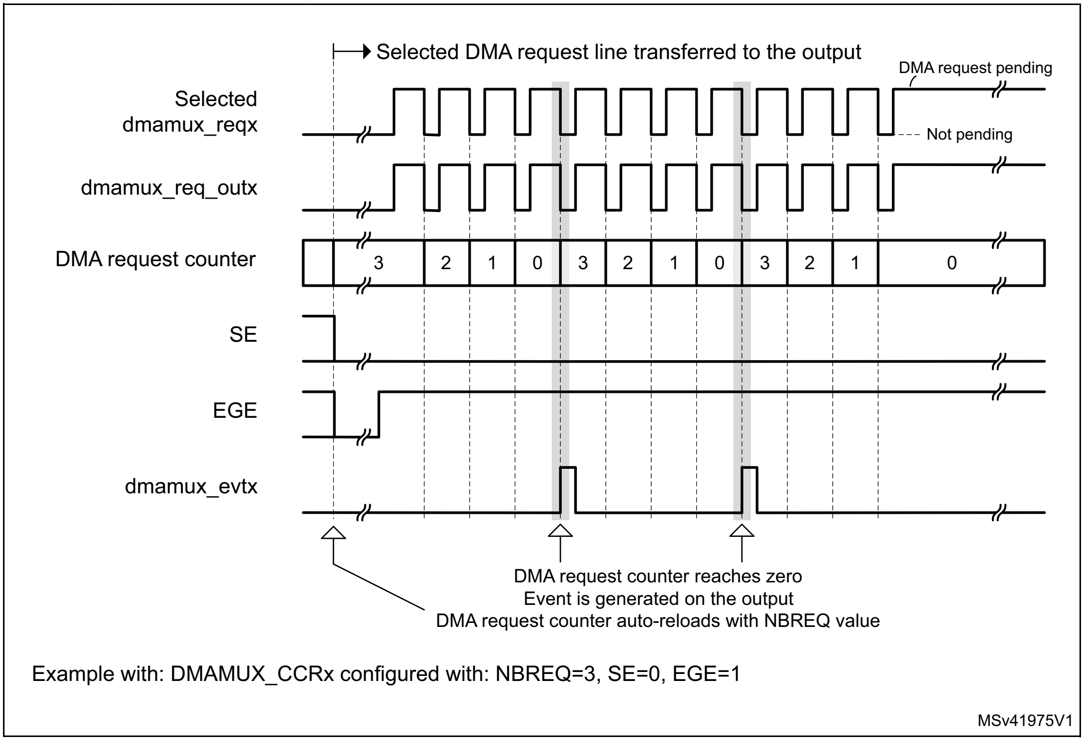

Selected DMA request line transferred to the output
DMA requests served
DMA request pending
Selected dmamux_reqx
Not pending
dmamux_syncx
dmamux_req_outx
DMA request counter
4
3
2
1
0
4
dmamux_evtx
Synchronization event
Input DMA request line connected to output
DMA request counter underrun
DMA request counter auto-reload to NBREQ
Input DMA request line disconnected from output
Example: DMAMUX_CCRx configured with: NBREQ=4, SE=1, EGE=1, SPOL=01 (rising edge)
MSv41974V1

Figure 34. Event generation of the DMA request line multiplexer channel


Selected DMA request line transferred to the output
DMA request pending
Selected dmamux_reqx
Not pending
dmamux_req_outx
DMA request counter
3
2
1
0
3
2
1
0
3
2
1
0
SE
EGE
dmamux_evtx
DMA request counter reaches zero
Event is generated on the output
DMA request counter auto-reloads with NBREQ value
Example with: DMAMUX_CCRx configured with: NBREQ=3, SE=0, EGE=1
MSv41975V1

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 33* and *Figure 34*.

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_CFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX_CxCR register.

13.4.5 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.

13.5 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 95. 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

13.6 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.

13.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)

Address offset: 0x000 + 0x04 * x (x = 0 to 15)

Reset value: 0x0000 0000

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	Res.	DMAREQ_ID[6:0]						
						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 *Table 93: DMAMUX: assignment of synchronization inputs to resources*).

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

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DMAREQ_ID[6:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

13.6.2 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
SOF15	SOF14	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	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SOF[15:0]**: 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 register.

13.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSOF 15	CSOF 14	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	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CSOF[15:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX_CSR register.

13.6.4 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)

Address offset: 0x100 + 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
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

13.6.5 DMAMUX request generator interrupt status register (DMAMUX_RGSR)

Address offset: 0x140

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.	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.

13.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)

Address offset: 0x144

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.	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.

13.6.7 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 96. 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	DMAMUX_C14CR	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
0x03C	DMAMUX_C15CR	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
0x040 - 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.

Table 96. 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
0x080	DMAMUX_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOF15	SOF14	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	0	0
0x084	DMAMUX_CCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSOF15	CSOF14	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	0	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 2.2 on page 81* for the register boundary addresses.

14 Nested vectored interrupt controller (NVIC)

14.1 NVIC main features

- 102 maskable interrupt channels (not including the sixteen Cortex®-M4 with FPU interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

The NVIC and the processor core interface are closely coupled, which enables 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 products.

14.2 SysTick calibration value register

The SysTick calibration value is set to 0x3E8, which gives a reference time base of 1 ms with the SysTick clock set to 125 KHz (1 MHz / 8).

14.3 Interrupt and exception vectors

The gray rows in *Table 97* describe the vectors without specific position.

Table 97. STM32G4 Series vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	Non maskable interrupt. SRAM parity err + FLASH ECC err + HSE CSS	0x0000 0008
-	-1	fixed	HardFault	All classes of fault	0x0000 000C
-	0	settable	MemManage	Memory management	0x0000 0010
-	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C - 0x0000 0028
-	3	-	-	System service call via SWI instruction	0x0000 002C
-	4	-	-	Monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	5	settable	PendSV	Pendable request for system service	0x0000 0038
-	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD_PVM	PVD through EXTI line 16 interrupt	0x0000 0044
2	9	settable	RTC/TAMP/CSS_LSE	RTC/TAMP/CSS on LSE through EXTI line 19 interrupt	0x0000 0048
3	10	settable	RTC_WKUP	RTC Wakeup timer through EXTI line 20 interrupt	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000 0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000 0068
11	18	settable	DMA1_CH1	DMA1 channel 1 interrupt	0x0000 006C
12	19	settable	DMA1_CH2	DMA1 channel 2 interrupt	0x0000 0070
13	20	settable	DMA1_CH3	DMA1 channel 3 interrupt	0x0000 0074

Table 97. STM32G4 Series vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
14	21	settable	DMA1_CH4	DMA1 channel 4 interrupt	0x0000 0078
15	22	settable	DMA1_CH5	DMA1 channel 5 interrupt	0x0000 007C
16	23	settable	DMA1_CH6	DMA1 channel 6 interrupt	0x0000 0080
17	24	settable	DMA1_CH7	DMA1 channel 7 interrupt	0x0000 0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000 0088
19	26	settable	USB_HP	USB High priority interrupts	0x0000 008C
20	27	settable	USB_LP	USB Low priority interrupts	0x0000 0090
21	28	settable	fdcan1_intr1_it	FDCAN1 interrupt 0	0x0000 0094
22	29	settable	fdcan1_intr0_it	FDCAN1 interrupt1	0x0000 0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM1_BRK/TIM15	TIM1 Break/TIM15 global interrupts	0x0000 00A0
25	32	settable	TIM1_UP/TIM16	TIM1 Update/TIM16 global interrupts	0x0000 00A4
26	33	settable	TIM1_TRG_COM/ TIM17/ TIM1_DIR/TIM1_IDX	TIM1 trigger and commutation/TIM17 interrupts/TIM1 Direction Change interrupt/TIM1 Index	0x0000 00A8
27	34	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000 00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000 00B8
31	38	settable	I2C1_EV	I2C1 event interrupt & EXTI line 23 interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000 00C0
33	40	settable	I2C2_EV	I2C2 event interrupt & EXTI line 24 interrupt	0x0000 00C4
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000 00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000 00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000 00D0
37	44	settable	USART1	USART1 global interrupt and EXTI line 25	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt and EXTI line 26	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt and EXTI line 28	0x0000 00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_ALARM	RTC alarms interrupts	0x0000 00E4
42	49	settable	USBWakeUP	USB wakeup from suspend (EXTI line 18)	0x0000 00E8
43	50	settable	TIM8_BRK/TIM8_TERR/TIM8_IERR	TIM8 Break interrupt/TIM8 Transition error/TIM8 Index error	0x0000 00EC
44	51	settable	TIM8_UP	TIM8 Update interrupt	0x0000 00F0

Table 97. STM32G4 Series vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
45	52	settable	TIM8_TRG_COM/TIM8_DIR/TIM8_IDX	TIM8 trigger and commutation interrupt/TIM8 Direction Change interrupt/TIM8 Index	0x0000 00F4
46	53	settable	TIM1_CC	TIM8 capture compare interrupt	0x0000 00F8
47	54	settable	ADC3	ADC3 global interrupt	0x0000 00FC
48	55	settable	FSMC	FSMC global interrupt	0x0000 0100
49	56	settable	LPTIM1	LPTIM1 global interrupt	0x0000 0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000 0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000 010C
52	59	settable	UART4	UART4 global interrupt and EXTI line 34 interrupts	0x0000 0110
53	60	settable	UART5	UART5 global interrupt and EXTI line 35 interrupts	0x0000 0114
54	61	settable	TIM6_DACUNDER	TIM6 and DAC1/3 underrun global interrupts	0x0000 0118
55	62	settable	TIM7_DACUNDER	TIM7 and DAC2/4 underrun global interrupts	0x0000 011C
56	63	settable	DMA2_CH1	DMA2 channel 1 interrupt	0x0000 0120
57	64	settable	DMA2_CH2	DMA2 channel 2 interrupt	0x0000 0124
58	65	settable	DMA2_CH3	DMA2 channel 3 interrupt	0x0000 0128
59	66	settable	DMA2_CH4	DMA2 channel 4 interrupt	0x0000 012C
60	67	settable	DMA2_CH5	DMA2 channel 5 interrupt	0x0000 0130
61	68	settable	ADC4	ADC4 global interrupt	0x0000 0134
62	69	settable	ADC5	ADC5 global interrupt	0x0000 0138
63	70	settable	UCPD1 global interrupt	UCPD1 global interrupt and EXTI line 43	0x0000 013C
64	71	settable	COMP1_2_3	COMP1/COMP2/COMP3 through EXTI lines 21/22/29 interrupts	0x0000 0140
65	72	settable	COMP4_5_6	COMP4/COMP5/COMP6 through EXTI lines 30/31/32 interrupts	0x0000 0144
66	73	settable	COMP7	lobal interrupt COMP7 through EXTI line 33	0x0000 0148
67	74	settable	HRTIM_Master_IRQn	HRTIM master timer interrupt (hrtim_it1)	0x0000 014C
68	75	settable	HRTIM_TIMA_IRQn	HRTIM timer A interrupt (hrtim_it2)	0x0000 0150
69	76	settable	HRTIM_TIMB_IRQn	HRTIM timer B interrupt (hrtim_it3)	0x0000 0154
70	77	settable	HRTIM_TIMC_IRQn	HRTIM timer C interrupt (hrtim_it4)	0x0000 0158
71	78	settable	HRTIM_TIMD_IRQn	HRTIM timer D interrupt (hrtim_it5)	0x0000 015C
72	79	settable	HRTIM_TIME_IRQn	HRTIM timer E interrupt (hrtim_it6)	0x0000 0160
73	80	settable	HRTIM_TIM_FLT_IRQn	HRTIM fault interrupt (hrtim_it8)	0x0000 0164

Table 97. STM32G4 Series vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
74	81	settable	HRTIM_TIMF_IRQn	hrtim_it7 / HRTIM timer F interrupt	0x0000 0168
75	82	settable	CRS	CRS interrupt	0x0000 016C
76	83	settable	SAI	SAI	0x0000 0170
77	84	settable	TIM20_BRK/ TIM20_TERR/ TIM20_IERR	TIM20 Break interrupt/TIM20 Transition error/TIM20 Index error	0x0000 0174
78	85	settable	TIM20_UP	TIM20 Update interrupt	0x0000 0178
79	86	settable	TIM20_TRG_COM/ TIM20_DIR/TIM20_IDX	TIM20 Trigger and commutation interrupt/TIM20 Direction Change interrupt/TIM20 Index	0x0000 017C
80	87	settable	TIM20_CC	TIM20 capture compare interrupt	0x0000 0180
81	88	settable	FPU	Floating point interrupt	0x0000 0184
82	89	settable	I2C4_EV	I2C4 event interrupt and EXTI line 42	0x0000_0188
83	90	settable	I2C4_ER	I2C4 error interrupt	0x0000_018C
84	91	settable	SPI4	SPI4 global interrupt	0x0000_0190
85	92	settable	AES	AES global interrupt	0x0000_0194
86	93	settable	FDCAN2_intr0	FDCAN2 Interrupt 0	0x0000_0198
87	94	settable	FDCAN2_intr1	FDCAN2 Interrupt 1	0x0000_019C
88	95	settable	FDCAN3_intr0	FDCAN3 Interrupt 0	0x0000_01A0
89	96	settable	FDCAN3_intr1	FDCAN3 Interrupt 0	0x0000_01A4
90	97	settable	RNG	RNG global interrupt	0x0000_01A8
91	98	settable	LPUART	LPUART global interrupt	0x0000_01AC
92	99	settable	I2C3_EV	I2C3 event and EXTI line 27 interrupts	0x0000_01B0
93	100	settable	I2C3_ER	I2C3 error interrupt	0x0000_01B4
94	101	settable	DMAMUX_OVR	DMAMUX Overrun interrupt	0x0000_01B8
95	102	settable	QUADSPI	QUADSPI global interrupt	0x0000_01BC
96	103	settable	DMA1_CH8	DMA1 channel 8 interrupt	0x0000_01C0
97	104	settable	DMA2_CH6	DMA2 channel 6 interrupt	0x0000_01C4
98	105	settable	DMA2_CH7	DMA2 channel 7 interrupt	0x0000_01C8
99	106	settable	DMA2_CH8	DMA2 channel 8 interrupt	0x0000_01CC
100	107	settable	Cordic	Cordic interrupt	0x0000_01D0
101	108	settable	FMAC	FMAC interrupt	0x0000_01D4

15 Extended interrupts and events controller (EXTI)

15.1 Introduction

The EXTI main features are as follows:

- Generation of up to 42 event/interrupt requests
  - 28 configurable lines
  - 14 direct lines
- Independent mask on each event/interrupt line
- Configurable rising or falling edge (configurable lines only)
- Dedicated status bit (configurable lines only)
- Emulation of event/interrupt requests (configurable lines only)

15.2 EXTI main features

The extended interrupts and events controller (EXTI) manages the external and internal asynchronous events/interrupts and generates the event request to the CPU/Interrupt Controller and a wake-up request to the Power Controller.

The EXTI allows the management of up to 42 event lines which can wake up from the Stop mode.

The lines are either configurable or direct:

- The lines are configurable: the active edge can be chosen independently, and a status flag indicates the source of the interrupt. The configurable lines are used by the I/Os external interrupts, and by few peripherals.
- The lines are direct: they are used by some peripherals to generate a wakeup from Stop event or interrupt. The status flag is provided by the peripheral.

Each line can be masked independently for an interrupt or an event generation.

This controller also allows to emulate events or interrupts by software, multiplexed with the corresponding hardware event line, by writing to a dedicated register.

15.3 EXTI functional description

For the configurable interrupt lines, the interrupt line should be configured and enabled in order to generate an interrupt. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1’ to the corresponding bit in the interrupt mask register. When the selected edge occurs on the interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is cleared by writing a ‘1’ in the pending register.

For the direct interrupt lines, the interrupt is enabled by default in the interrupt mask register and there is no corresponding pending bit in the pending register.

To generate an event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a ‘1’ to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set.

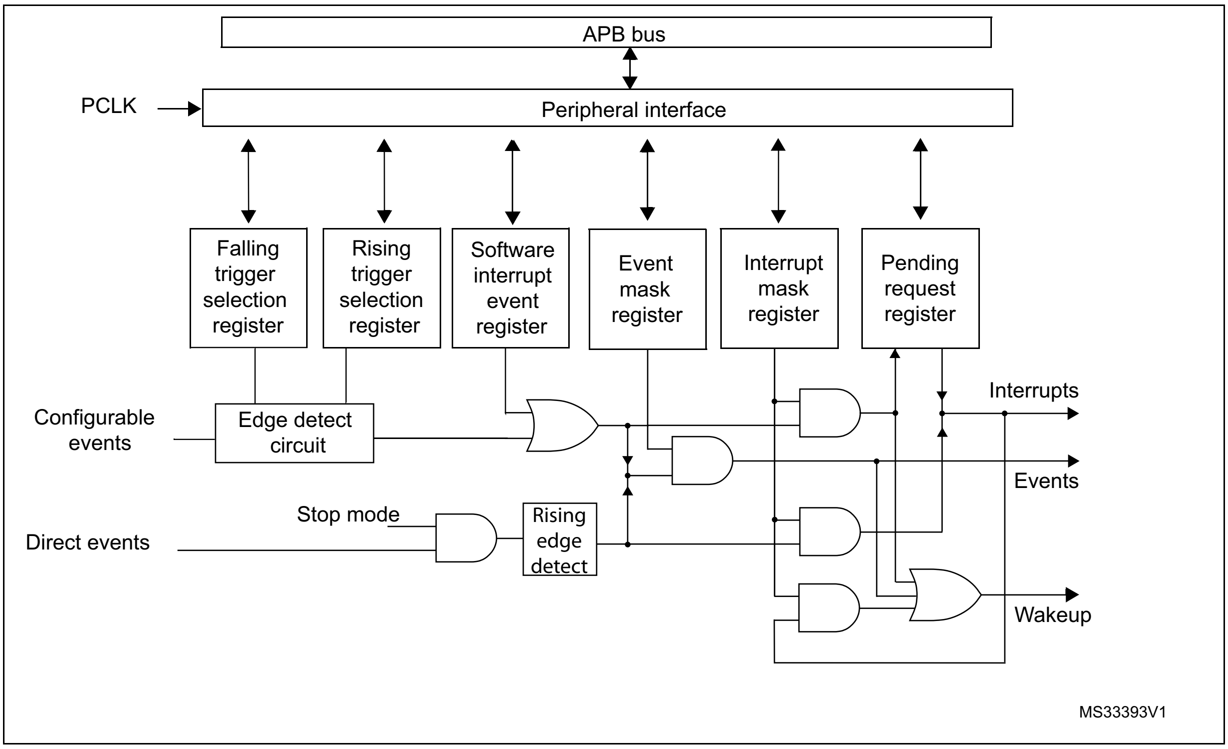
For the configurable lines, an interrupt/event request can also be generated by software by writing a '1' in the software interrupt/event register.

Note: The interrupts or events associated to the direct lines are triggered only when the system is in Stop mode. If the system is still running, no interrupt/event is generated by the EXTI.

15.3.1 EXTI block diagram

The extended interrupt/event block diagram is shown on *Figure 35*.

Figure 35. Configurable interrupt/event block diagram



15.3.2 Wakeup event management

The STM32G4 Series is able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated either by:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex^{TM}-M4 System Control register. When the MCU resumes from WFE, the EXTI peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared
- or by configuring an EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

15.3.3 Peripherals asynchronous Interrupts

Some peripherals are able to generate events when the system is in run mode and also when the system is in Stop mode, allowing to wake up the system from Stop mode.

To accomplish this, the peripheral generates both a synchronized (to the system clock, e.g. APB clock) and an asynchronous version of the event. This asynchronous event is connected to an EXTI direct line.

Note: Few peripherals with wakeup from Stop capability are connected to an EXTI configurable line. In this case, the EXTI configuration is necessary to allow the wakeup from Stop mode.

15.3.4 Hardware interrupt selection

To configure a line as an interrupt source, use the following procedure:

1. Configure the corresponding mask bit in the EXTI_IMR register.
2. Configure the Trigger Selection bits of the Interrupt line (EXTI_RTSR and EXTI_FTSR).
3. Configure the enable and mask bits that control the NVIC IRQ channel mapped to the EXTI so that an interrupt coming from one of the EXTI lines can be correctly acknowledged.

Note: The direct lines do not require any EXTI configuration.

15.3.5 Hardware event selection

To configure a line as an event source, use the following procedure:

1. Configure the corresponding mask bit in the EXTI_EMR register.
2. Configure the Trigger Selection bits of the Event line (EXTI_RTSR and EXTI_FTSR).

15.3.6 Software interrupt/event selection

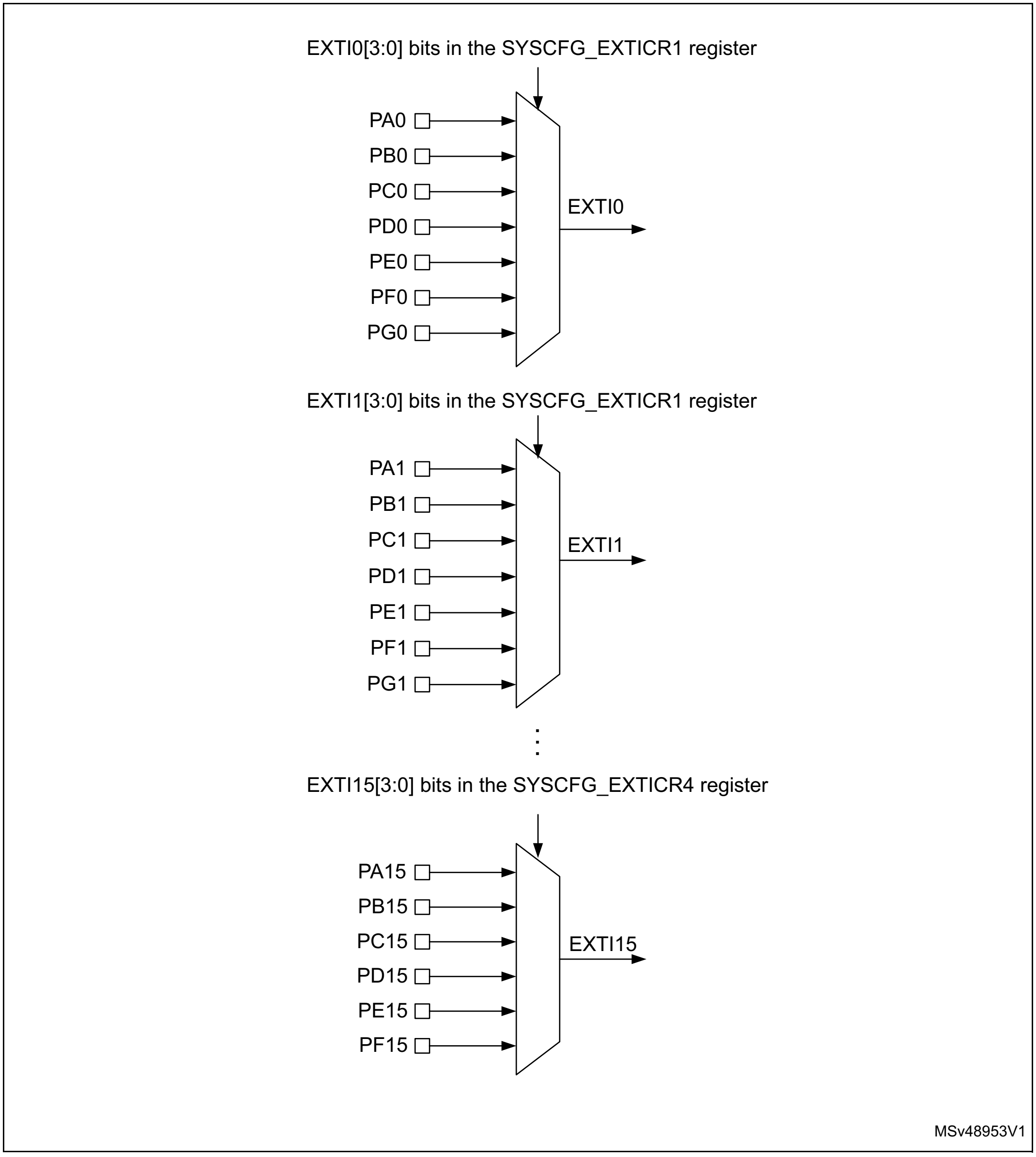
Any of the configurable lines can be configured as a software interrupt/event line. The procedure to generate a software interrupt is as follows:

1. Configure the corresponding mask bit (EXTI_IMR, EXTI_EMR).
2. Set the required bit of the software interrupt register (EXTI_SWIER).

15.4 EXTI interrupt/event line mapping

In the STM32G4 Series, 42 interrupt/event lines are available. The GPIOs are connected to 16 configurable interrupt/event lines (see *Figure 36*).

Figure 36. External interrupt/event GPIO mapping


EXTI0[3:0] bits in the SYSCFG_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PF0
PG0
EXTI0
EXTI1[3:0] bits in the SYSCFG_EXTICR1 register
PA1
PB1
PC1
PD1
PE1
PF1
PG1
EXTI1
EXTI15[3:0] bits in the SYSCFG_EXTICR4 register
PA15
PB15
PC15
PD15
PE15
PF15
EXTI15
MSv48953V1

The EXTI lines are connected as shown in *Table 98: EXTI lines connections*.

Table 98. EXTI lines connections

EXTI line	Line source	Line type
0-15	GPIO	configurable
16	PVD	configurable
17	RTC alarm event	configurable
18	USB Device FS wakeup event	direct
19	Timestamp or CSS_LSE	configurable
20	RTC wakeup timer	configurable

Table 98. EXTI lines connections (continued)

EXTI line	Line source	Line type
21	COMP1 output	configurable
22	COMP2 output	configurable
23	I2C1 wakeup	direct
24	I2C2 wakeup	direct
25	USART1 wakeup	direct
26	USART2 wakeup	direct
27	I2C3 wakeup	direct
28	USART3 wakeup	direct
29	COMP3 output	configurable
30	COMP4 output	configurable
31	COMP5 output	configurable
32	COMP6 output	configurable
33	COMP7 output	configurable
34	UART4 wakeup	direct
35	UART5 wakeup	direct
36	LPUART1 wakeup	direct
37	LPTIM1 wakeup	direct
40	PVM1 wakeup	configurable
41	PVM2 wakeup	configurable
42	I2C4 wakeup	direct
43	UCPD1	direct

15.5 EXTI registers

Refer to *Section 1.2 on page 73* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

15.5.1 Interrupt mask register 1 (EXTI_IMR1)

Address offset: 0x00

Reset value: Direct lines are set to '1', others lines are set to '0'. See *Table 98: EXTI lines connections*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
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
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IMx:** Interrupt Mask on line x (x = 31 to 0)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is not masked

Note: The reset value for the direct lines is set to '1' in order to enable the interrupt by default.

15.5.2 Event mask register 1 (EXTI_EMR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
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
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 31:0 **EMx:** Event mask on line x (x = 31 to 0)
0: Event request from line x is masked
1: Event request from line x is not masked

15.5.3 Rising trigger selection register 1 (EXTI_RTSR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RT31	RT30	RT29	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	RT20	RT19	Res.	RT17	RT16
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
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 31:29 **RTx:** Rising trigger event configuration bit of line x (x = 31 to 29)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:19 **RTx:** Rising trigger event configuration bit of line x (x = 22 to 19)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **RTx:** Rising trigger event configuration bit of line x (x = 17 to 0)
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 wakeup lines are edge-triggered. No glitch must be generated on these lines. If a rising edge on a configurable interrupt line occurs during a write operation in the EXTI_RTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

15.5.4 Falling trigger selection register 1 (EXTI_FTSR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FT31	FT30	FT29	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	FT20	FT19	Res.	FT17	FT16
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
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 31:29 **FTx:** Falling trigger event configuration bit of line x (x = 31 to 29)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:19 **FTx:** Falling trigger event configuration bit of line x (x = 22 to 19)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **FTx:** Falling trigger event configuration bit of line x (x = 17 to 0)
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 wakeup lines are edge-triggered. No glitch must be generated on these lines. If a falling edge on a configurable interrupt line occurs during a write operation to the EXTI_FTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

15.5.5 Software interrupt event register 1 (EXTI_SWIER1)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:29 **SWIx:** Software interrupt on line x (x = 31 o 29)
If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.
This bit is cleared by clearing the corresponding bit in the EXTI_PR register (by writing a ‘1’ into the bit).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22: 19 **SWIx:** Software interrupt on line x (x = 22 o 19)

If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit in the EXTI_PR register (by writing a ‘1’ into the bit).

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **SWIx:** Software interrupt on line x (x = 17 to 0)

If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a ‘1’ into the bit).

15.5.6 Pending register 1 (EXTI_PR1)

Address offset: 0x14

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIF31	PIF30	PIF29	Res.	Res.	Res.	Res.	Res.	Res.	PIF22	PIF21	PIF20	PIF19	Res.	PIF17	PIF16
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
PIF15	PIF14	PIF13	PIF12	PIF11	PIF10	PIF9	PIF8	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
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 31:29 **PIFx:** Pending interrupt flag on line x (x = 31 to 29)

0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a ‘1’ to the bit.

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:19 **PIFx:** Pending interrupt flag on line x (x = 22 to 19)

0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a ‘1’ to the bit.

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **PIFx:** Pending interrupt flag on line x (x = 17 to 0)

0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a ‘1’ to the bit.

15.5.7 Interrupt mask register 2 (EXTI_IMR2)

Address offset: 0x20

Reset value: Direct lines are set to '1', others lines are set to '0'. See *Table 98: EXTI lines connections*.

31	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.	IM43	IM42	IM41	IM40	Res.	Res.	IM37	IM36	IM35	IM34	IM33	IM32
				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value

Bits 11:8 **IMx:** Interrupt mask on line x (x = 43 to 40)
0: Interrupt request from line x is masked
1: Interrupt request from line x is not masked

Bits 7:6 Reserved, must be kept at reset value

Bits 5:0 **IMx:** Interrupt mask on line x (x = 37 to 32)
0: Interrupt request from line x is masked
1: Interrupt request from line x is not masked

Note: The reset value for the direct lines is set to '1' in order to enable the interrupt by default.

15.5.8 Event mask register 2 (EXTI_EMR2)

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.	EM43	EM42	EM41	EM40	Res.	Res.	EM37	EM36	EM35	EM34	EM33	EM32
				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value

Bits 11:8 **EMx:** Event mask on line x (x = 43 to 40)
0: Event request from line x is masked
1: Event request from line x is not masked

Bits 7:6 Reserved, must be kept at reset value

Bits 5:0 **EMx:** Event mask on line x (x = 37 to 32)
0: Event request from line x is masked
1: Event request from line x is not masked

15.5.9 Rising trigger selection register 2 (EXTI_RTSR2)

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.	RT41	RT40	Res.	Res.	Res.	Res.	Res.	Res.	RT33	RT32
						rw	rw							rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **RTx:** Rising trigger event configuration bit of line x (x = 40 to 41)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

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

Bits 1:0 **RTx:** Rising trigger event configuration bit of line x (x = 32 to 33)
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 wakeup lines are edge-triggered. No glitch must be generated on these lines. If a rising edge on a configurable interrupt line occurs during a write operation to the EXTI_RTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

15.5.10 Falling trigger selection register 2 (EXTI_FTSR2)

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.	FT41	FT40	Res.	Res.	Res.	Res.	Res.	Res.	FT33	FT32
						rw	rw							rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **FTx:** Falling trigger event configuration bit of line x (x = 40 to 41)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line

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

Bits 1:0 **FTx:** Falling trigger event configuration bit of line x (x = 32 to 33)
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 wakeup lines are edge-triggered. No glitch must be generated on these lines. If a falling edge on a configurable interrupt line occurs during a write operation to the EXTI_FTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

15.5.11 Software interrupt event register 2 (EXTI_SWIER2)

Address offset: 0x30

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.	SWI41	SWI40	Res.	Res.	Res.	Res.	Res.	Res.	SWI33	SWI32
						rw	rw							rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **SWIx:** Software interrupt on line x (x = 40 to 41)

If the interrupt is enabled on this line in EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit of EXTI_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a ‘1’ to the bit).

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

Bits 1:0 **SWIx:** Software interrupt on line x (x = 32 to 33)

If the interrupt is enabled on this line in EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit of EXTI_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a ‘1’ to the bit).

15.5.12 Pending register 2 (EXTI_PR2)

Address offset: 0x34

Reset value: undefined

31	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.	PIF41	PIF40	Res.	Res.	Res.	Res.	Res.	Res.	PIF33	PIF32
						rc_w1	rc_w1							rc_w1	rc_w1

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **PIFx:** Pending interrupt flag on line x (x = 40 to 41)

0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a ‘1’ into the bit.

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

Bits 1:0 **PIFx:** Pending interrupt flag on line x (x = 32 to 33)

0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a ‘1’ into the bit.

15.5.13 EXTI register map

Table 99 gives the EXTI register map and the reset values.

Table 99. Extended interrupt/event controller 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	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	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	EXTI_EMR1	EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16	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	0	0	0	0	0	0	0	0	0	0
0x08	EXTI_RTSR1	RT31	RT30	RT29	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	RT20	RT19	Res.	RT17	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	0	0	0	0	0	0
0x0C	EXTI_FTSR1	FT31	FT30	FT29	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	FT20	FT19	Res.	FT17	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	0	0	0	0	0	0
0x10	EXTI_SWIER1	SW312	SWI30	SWI29	Res.	Res.	Res.	Res.	Res.	Res.	SWI22	SWI21	SWI20	SWI19	Res.	SWI17	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	0	0	0	0	0	0
0x14	EXTI_PR1	PIF31	PIF30	PIF29	Res.	Res.	Res.	Res.	Res.	Res.	PIF22	PIF21	PIF20	PIF19	Res.	PIF17	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	0	0	0	0	0	0
0x20	EXTI_IMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM43	IM42	IM41	IM40	Res.	Res.	IM37	IM36	IM35	IM34	IM33	IM32
	Reset value																					1	1	0	1			0	0	0	1	1	1
0x24	EXTI_EMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM43	EM42	EM41	EM40	Res.	Res.	EM37	EM36	EM35	EM34	EM33	EM32
	Reset value																					0	0	0	0			0	0	0	0	0	0
0x28	EXTI_RTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT41	RT40	Res.	Res.	Res.	Res.	Res.	Res.	RT33	RT32
	Reset value																							0	0							0	0
0x2C	EXTI_FTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT41	FT40	Res.	Res.	Res.	Res.	Res.	Res.	FT33	FT32
	Reset value																							0	0							0	0
0x30	EXTI_SWIER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI41	SWI40	Res.	Res.	Res.	Res.	Res.	Res.	SWI33	SWI32
	Reset value																							0	0							0	0
0x34	EXTI_PR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF41	PIF40	Res.	Res.	Res.	Res.	Res.	Res.	PIF33	PIF32
	Reset value																							0	0							0	0

Refer to *Section 2.2 on page 81* for the register boundary addresses.

16 Cyclic redundancy check calculation unit (CRC)

16.1 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.

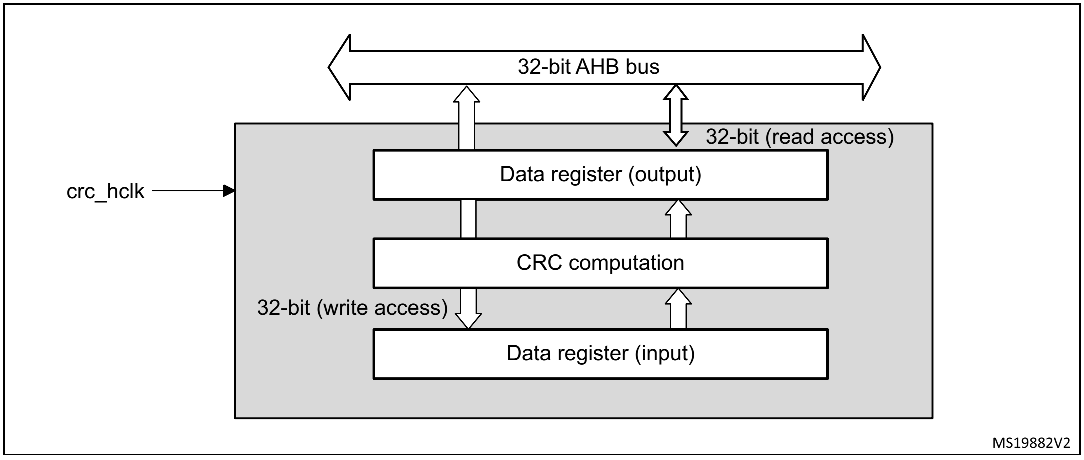
16.2 CRC main features

- 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

16.3 CRC functional description

16.3.1 CRC block diagram

Figure 37. CRC calculation unit block diagram



16.3.2 CRC internal signals

Table 100. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

16.3.3 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 access is allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32-bit
- 2 AHB clock cycles for 16-bit
- 1 AHB clock cycles for 8-bit

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: input data 0x1A2B3C4D is 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, output data 0x11223344 is converted into 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.

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.

16.4 CRC registers

16.4.1 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.

16.4.2 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

16.4.3 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

These bits control 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

16.4.4 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.

16.4.5 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.

16.4.6 CRC register map

Table 101. 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 2.2 on page 81* for the register boundary addresses.

17 CORDIC co-processor (CORDIC)

17.1 CORDIC introduction

The CORDIC co-processor provides hardware acceleration of mathematical functions (mainly trigonometric ones) commonly used in motor control, metering, signal processing and many other applications.

It speeds up the calculation of these functions compared to a software implementation, making it possible the use of a lower operating frequency, or freeing up processor cycles in order to perform other tasks.

17.2 CORDIC main features

- 24-bit CORDIC rotation engine
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Functions: sine, cosine, sinh, cosh, atan, atan2, atanh, modulus, square root, natural logarithm
- Programmable precision
- Low latency AHB slave interface
- Results can be read as soon as ready, without polling or interrupt
- DMA read and write channels
- Multiple register read/write by DMA

17.3 CORDIC functional description

17.3.1 General description

The CORDIC is a cost-efficient successive approximation algorithm for evaluating trigonometric and hyperbolic functions.

In trigonometric (circular) mode, the sine and cosine of an angle θ are determined by rotating the unit vector [1, 0] through decreasing angles until the cumulative sum of the rotation angles equals the input angle θ. The x and y cartesian components of the rotated vector then correspond, respectively, to the cosine and sine of θ. Inversely, the angle of a vector [x, y] corresponding to arctangent (y / x), is determined by rotating [x, y] through successively decreasing angles to obtain the unit vector [1, 0]. The cumulative sum of the rotation angles gives the angle of the original vector.

The CORDIC algorithm can also be used for calculating hyperbolic functions (sinh, cosh, atanh), by replacing the successive circular rotations by steps along a hyperbole.

Other functions can be derived from the basic functions described above.

17.3.2 CORDIC functions

The first step when using the co-processor is to select the required function, by programming the FUNC field of the CORDIC_CR register accordingly.

Table 102 lists the functions supported by the CORDIC co-processor.

Table 102. CORDIC functions

Function	Primary argument (ARG1)	Secondary argument (ARG2)	Primary result (RES1)	Secondary result (RES2)
Cosine	angle θ	modulus m	$m \cdot \cos\theta$	$m \cdot \sin\theta$
Sine	angle θ	modulus m	$m \cdot \sin\theta$	$m \cdot \cos\theta$
Phase	x	y	atan2(y,x)	$\sqrt{x^2 + y^2}$
Modulus	x	y	$\sqrt{x^2 + y^2}$	atan2(y,x)
Arctangent	x	none	$\tan^{-1} x$	none
Hyperbolic cosine	x	none	cosh x	sinh x
Hyperbolic sine	x	none	sinh x	cosh x
Hyperbolic arctangent	x	none	$\tanh^{-1} x$	none
Natural logarithm	x	none	ln x	none
Square root	x	none	$\sqrt{x}$	none

Several functions take two input arguments (ARG1 and ARG2) and some generate two results (RES1 and RES2) simultaneously. This is a side-effect of the algorithm and means that only one operation is needed to obtain two values. This is the case, for example, when performing polar-to-rectangular conversion: sin θ also generates cos θ, cos θ also generates sin θ. Similarly for rectangular-to-polar conversion (phase(x,y), modulus(x,y)) and for hyperbolic functions (cosh θ, sinh θ).

Note: *The exponential function, exp x, can be obtained as the sum of sinh x and cosh x. Furthermore, base N logarithms, $\log_N$ x, can be derived by multiplying ln x by a constant K, where K = 1/ln N.*

For certain functions (atan, log, sqrt) a scaling factor (see Section 17.3.4) can be applied to extend the range of the function beyond the maximum [-1, 1] supported by the q1.31 fixed point format. The scaling factor must be set to 0 for all other circular functions, and to 1 for hyperbolic functions.

Cosine

Table 103. Cosine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	$m \cdot \cos\theta$	[-1, 1]

Table 103. Cosine parameters (continued)

Parameter	Description	Range
RES2	m · sin θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the cosine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the sine of the angle, multiplied by the modulus.

Sine

Table 104. Sine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m · sin θ	[-1, 1]
RES2	m · cos θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the sine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the sine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.

Phase

Table 105. Phase parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Phase angle θ in radians, divided by π	[-1, 1]

Table 105. Phase parameters (continued)

Parameter	Description	Range
RES2	Modulus m	[0, 1]
SCALE	Not applicable	0

This function calculates the phase angle in the range -π to π of a vector **v** = [x y] (also known as atan2(y,x). It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the phase angle θ of the vector **v**. RES1 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

The secondary result, RES2, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES2 is saturated to 1.

Modulus

Table 106. Modulus parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Modulus m	[0, 1]
RES2	Phase angle θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the magnitude, or modulus, of a vector **v** = [x y]. It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES1 is saturated to 1.

The secondary result, RES2, is the phase angle θ of the vector **v**. RES2 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

Arctangent

Table 107. Arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-1, 1]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \tan^{-1} x$, in radians, divided by p	[-1, 1]
RES2	Not applicable	-
SCALE	n	[0 7]

This function calculates the arctangent, or inverse tangent, of the input argument x.

The primary argument, ARG1, is the input value, $x = \tan \theta$. If $|x| > 1$, a scaling factor of 2^{-n} must be applied in software such that $-1 < x \cdot 2^{-n} < 1$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the scale factor n must be programmed in the SCALE parameter.

Note that the maximum input value allowed is $\tan \theta = 128$, which corresponds to an angle $\theta = 89.55$ degrees. For $|x| > 128$, a software method must be used to find $\tan^{-1} x$.

The secondary argument, ARG2, is unused.

The primary result, RES1, is the angle $\theta = \tan^{-1} x$. RES1 must be multiplied by $2^n \cdot \pi$ to obtain the angle in radians.

The secondary result, RES2, is unused.

Hyperbolic cosine

Table 108. Hyperbolic cosine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \cosh x$	[0.5 0.846]
RES2	$2^{-n} \cdot \sinh x$	[-0.683 0.683]
SCALE	n	1

This function calculates the hyperbolic cosine of a hyperbolic angle x. It can also be used to calculate the exponential functions $e^x = \cosh x + \sinh x$, and $e^{-x} = \cosh x - \sinh x$.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. Since the minimum value of cosh x is 1, which is beyond the range of the q1.31 format, a scaling factor of 2^{-n} must be applied in software. The factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic cosine, cosh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic sine, sinh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic sine

Table 109. Hyperbolic sine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559, 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \sinh x$	[-0.683, 0.683]
RES2	$2^{-n} \cdot \cosh x$	[0.5, 0.846]
SCALE	n	1

This function calculates the hyperbolic sine of a hyperbolic angle x. It can also be used to calculate the exponential functions e^x = cosh x + sinh x, and e^{-x} = cosh x- sinh x.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. For all input values, a scaling factor of 2^{-n} must be applied in software, where n = 1. The scaled value x · 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic sine, sinh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic cosine, cosh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic arctangent

Table 110. Hyperbolic arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.403 0.403]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot$atanh x	[-0.559 0.559]
RES2	Not applicable	-
SCALE	n	1

This function calculates the hyperbolic arctangent of the input argument x.

The primary argument is the input value x. Only values of x in the range -0.806 to +0.806 are supported. The value x must be scaled by a factor 2^{-n}, where n = 1. The scaled value

$x \cdot 0.5$ is programmed in ARG1 and the factor $n = 1$ must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result is the hyperbolic arctangent, atanh x. RES1 must be multiplied by 2 to obtain the correct value.

The secondary result is not used.

Natural logarithm

Table 111. Natural logarithm parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.054 0.875]
ARG2	Not applicable	-
RES1	$2^{-(n+1)}$.ln x	[-0.279 0.137]
RES2	Not applicable	-
SCALE	n	[1 4]

This function calculates the natural logarithm of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.107 to 9.35 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < 1-2^{-n}$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 112 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 112. Natural log scaling factors and corresponding ranges

n	x range	ARG1 range
1	$0.107 \leq x < 1$	$0.0535 \leq \text{ARG1} < 0.5$
2	$1 \leq x < 3$	$0.25 \leq \text{ARG1} < 0.75$
3	$3 \leq x < 7$	$0.375 \leq \text{ARG1} < 0.875$
4	$7 \leq x \leq 9.35$	$0.4375 \leq \text{ARG1} < 0.584$

The secondary argument is not used.

The primary result is the natural logarithm, ln x. RES1 must be multiplied by $2^{(n+1)}$ to obtain the correct value.

The secondary result is not used.

Square root

Table 113. Square root parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.027 0.875]
ARG2	Not applicable	-
RES1	$2^{-n}\sqrt{x}$	[0.04 1]
RES2	Not applicable	-
SCALE	n	[0 2]

This function calculates the square root of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.027 to 2.34 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < (1 - 2^{(-n-2)})$.

The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 114 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 114. Square root scaling factors and corresponding ranges

n	x range	ARG1 range
0	$0.027 \leq x < 0.75$	$0.027 \leq \text{ARG1} < 0.75$
1	$0.75 \leq x < 1.75$	$0.375 \leq \text{ARG1} < 0.875$
2	$1.75 \leq x \leq 2.341$	$0.4375 \leq \text{ARG1} \leq 0.585$

The secondary argument is not used.

The primary result is the square root of x. RES1 must be multiplied by 2^n to obtain the correct value.

The secondary result is not used.

17.3.3 Fixed point representation

The CORDIC operates in fixed point signed integer format. Input and output values can be either q1.31 or q1.15.

In q1.31 format, numbers are represented by one sign bit and 31 fractional bits (binary decimal places). The numeric range is therefore -1 (0x80000000) to $1 - 2^{-31}$ (0x7FFFFFFF).

In q1.15 format, the numeric range is 1 (0x8000) to $1 - 2^{-15}$ (0x7FFF). This format has the advantage that two input arguments can be packed into a single 32-bit write, and two results can be fetched in one 32-bit read.

17.3.4 Scaling factor

Several of the functions listed in *Section 17.3.2* specify a scaling factor, SCALE. This allows the function input range to be extended to cover the full range of values supported by the CORDIC, without saturating the input, output or internal registers. If the scaling factor is required, it has to be calculated in software and programmed into the SCALE field of the CORDIC_CSR register. The input arguments must be scaled accordingly before programming the scaled values in the CORDIC_WDATA register. The scaling must also be undone on the results read from the CORDIC_RDATA register.

Note: The scaling factor entails a loss of precision due to truncation of the scaled value.

17.3.5 Precision

The precision of the result is dependent on the number of CORDIC iterations. The algorithm converges at a constant rate of one binary digit per iteration for trigonometric functions (sine, cosine, phase, modulus), see *Figure 38*.

For hyperbolic functions (hyperbolic sine, hyperbolic cosine, natural logarithm), the convergence rate is less constant due to the peculiarities of the CORDIC algorithm (see *Figure 39*). The square root function converges at roughly twice the speed of the hyperbolic functions (see *Figure 40*).

Figure 38. CORDIC convergence for trigonometric functions

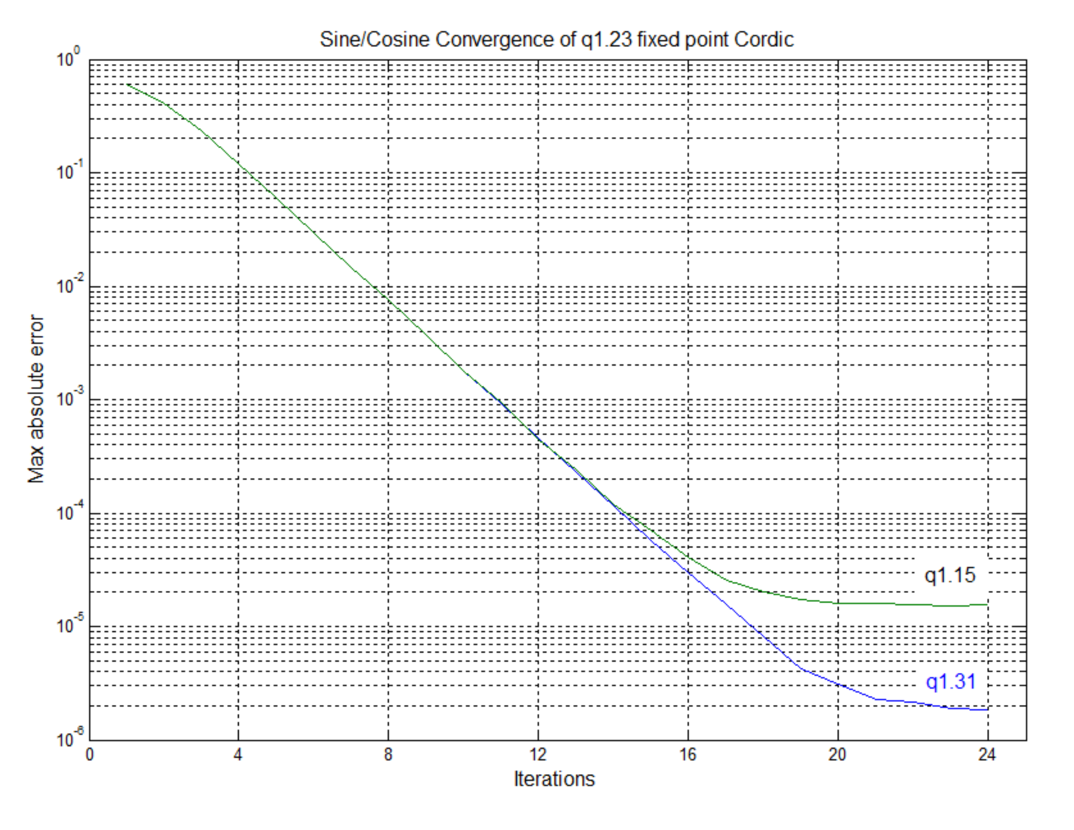


Figure 39. CORDIC convergence for hyperbolic functions

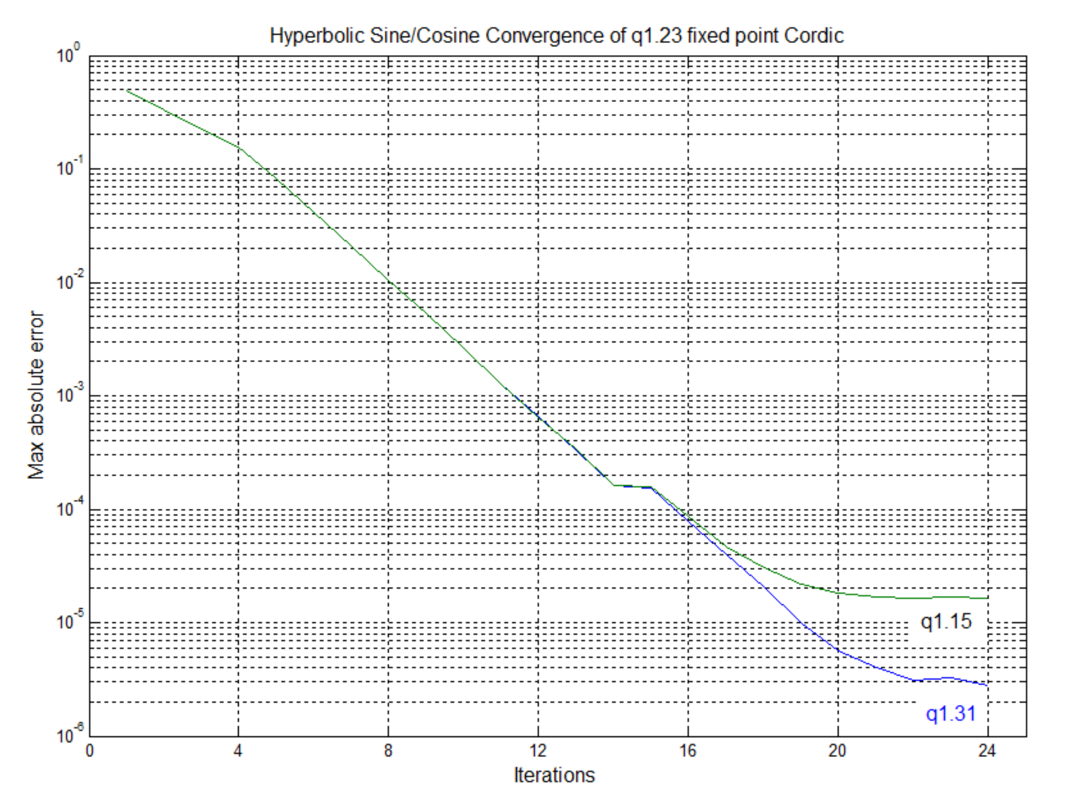
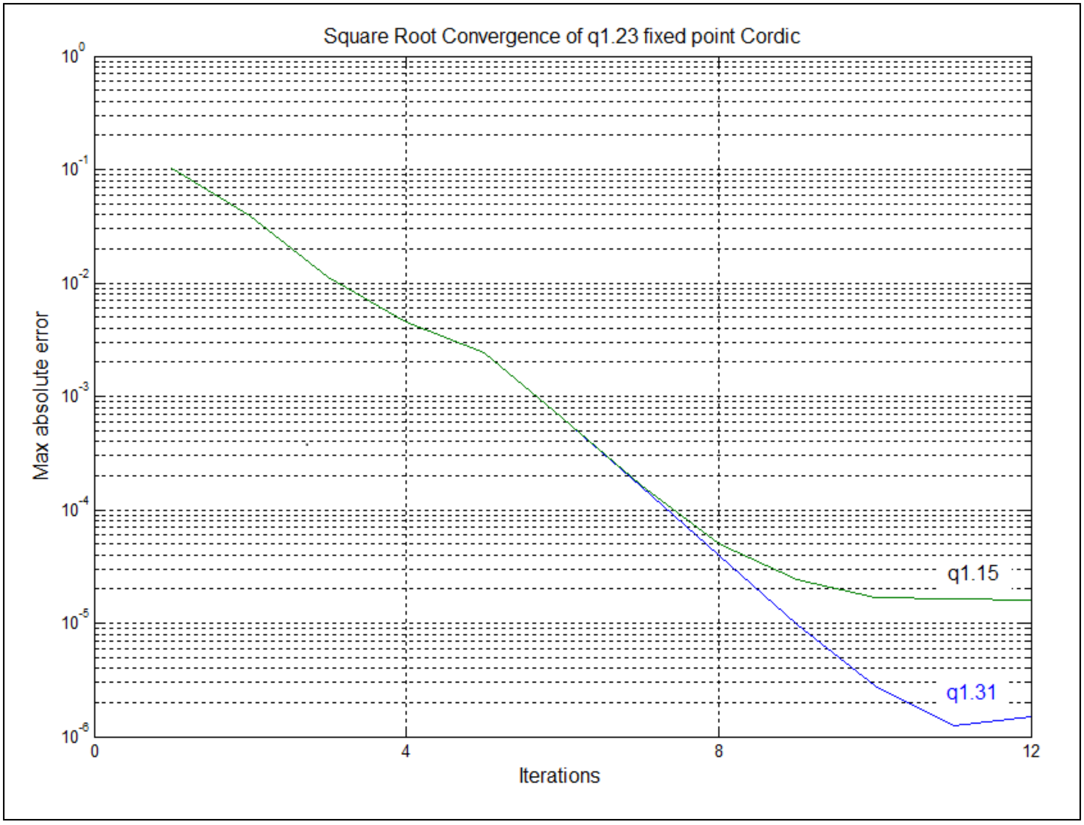


Figure 40. CORDIC convergence for square root



Note: The convergence rate decreases as the quantization error starts to become significant.

The CORDIC can perform four iterations per clock cycle. For each function, the maximum error remaining after every four iterations is shown in *Table 115*, together with the number of clock cycles required to reach that precision. From this table, the desired number of cycles can be determined and programmed in the PRECISION field of the CORDIC_CR register. The co-processor stops as soon as the programmed number of iterations has completed, and the result can be read immediately.

Table 115. Precision vs. number of iterations

Function	Number of iterations	Number of cycles	Max residual error[(1)]	
			q1.31 format	q1.15 format
Sin, Cos, Phase[(2)], Mod, Atan[(4)]	4	1	2^{-3}	2^{-3}
	8	2	2^{-7}	2^{-7}
	12	3	2^{-11}	2^{-11}
	16	4	2^{-15}	2^{-15}
	20	5	2^{-18}	2^{-16}
	24	6	2^{-19}	2^{-16}

Table 115. Precision vs. number of iterations (continued)

Function	Number of iterations	Number of cycles	Max residual error[1]	
			q1.31 format	q1.15 format
Sinh, Cosh, Atanh, Ln[3]	4	1	2^{-2}	2^{-2}
	8	2	2^{-6}	2^{-6}
	12	3	2^{-10}	2^{-10}
	16	4	2^{-13}	2^{-13}
	20	5	2^{-17}	2^{-15}
	24	6	2^{-18}	2^{-15}
Sqrt[4]	4	1	2^{-7}	2^{-7}
	8	2	2^{-14}	2^{-14}
	12	3	2^{-19}	2^{-15}

1. Max residual error is the maximum error remaining after the given number of iterations, compared to the identical calculation performed in double precision floating point. An additional rounding error may be incurred, of up to 2^{-16} for q15 format or 2^{-20} for q31 format.
2. For modulus > 0.5. The achievable precision reduces proportionally to the magnitude of the modulus, as quantization error becomes significant.
3. SCALE = 1. If a higher scaling factor is used, the achievable precision is reduced proportionally.
4. SCALE = 0. If a higher scaling factor is used, the achievable precision is reduced proportionally.

17.3.6 Zero-overhead mode

The fastest way to use the co-processor is to pre-program the CORDIC_CSR register with the function to be performed (FUNC), the desired number of clock cycles (PRECISION), the size of the input and output values (ARGSIZE, RESSIZE), the number of input arguments (NARGS) and/or results (NRES), and the scaling factor (SCALE), if applicable.

Subsequently, a calculation is triggered by writing the input arguments to the CORDIC_WDATA register. As soon as the correct number of input arguments has been written (and any ongoing calculation has finished) a new calculation is launched using these input arguments and the current CORDIC_CSR settings. There is no need to re-program the CORDIC_CSR register if there is no change.

If a dual 32-bit input argument is needed (ARGSIZE = 0, NARGS = 1), the primary input argument, ARG1, must be written first, followed by the secondary argument, ARG2. If the secondary argument remains unchanged for a series of calculations, the second write can be avoided, by reprogramming the number of arguments to one (NARGS = 0), once the first calculation has started. The secondary argument retains its programmed value as long as the function is not changed.

Note: ARG2 is set to +1 (0x7FFFFFFF) after a reset.

If two 16-bit arguments are used (ARGSIZE = 1) they must be packed into a 32-bit word, with ARG1 in the least significant half-word and ARG2 in the most significant half-word. The packed 32-bit word is then written to the CORDIC_WDATA register. Only one write is needed in this case (NARGS = 0).

For functions taking only one input argument, ARG1, it is recommended to set NARGS = 0. If NARGS = 1, a second write to CORDIC_WDATA must be performed to trigger the calculation. The ARG2 data in this case is not used.

Once the calculation starts, any attempt to read the CORDIC_RDATA register inserts bus wait states until the calculation is completed, before returning the result. Hence it is possible for the software to write the input and immediately read the result without polling to see if it is valid. Alternatively, the processor can wait for the appropriate number of clock cycles before reading the result. This time can be used to program the CORDIC_CSR register for the next calculation and prepare the next input data, if needed. The CORDIC_CSR register can be re-programmed while a calculation is in progress, without affecting the result of the ongoing calculation. In the same way, the CORDIC_WDATA register can be updated with the next argument(s) once the previous arguments have been taken into account. The next arguments and settings remain pending until the previous calculation has completed.

When a calculation is finished, the result(s) can be read from the CORDIC_RDATA register. If two 32-bit results are expected (NRES = 1, RESSIZE = 0), the primary result (RES1) is read out first, followed by the secondary result (RES2). If only one 32-bit result is expected (NRES = 0, RESSIZE = 0), then RES1 is output on the first read.

If 16-bit results are expected (RESSIZE = 1), a single read to CORDIC_RDATA fetches both results packed into a 32-bit word. RES1 is in the lower half-word, and RES2 in the upper half-word. In this case, it is recommended to program NRES = 0. IF NRES = 1, a second read of CORDIC_RDATA must be performed in order to free up the CORDIC for the next operation. The data from this second read must be discarded.

The next calculation starts when the expected number of results has been read, provided the expected number of arguments has been written. This means that at any time, there can be one calculation in progress, or waiting for the results to be read, and one operation pending. Any further access to CORDIC_WDATA while an operation is pending, cancels the pending operation and overwrite the data.

The following sequence summarizes the use of the CORDIC_IP in zero-overhead mode:

1. Program the CORDIC_CSR register with the appropriate settings
2. Program the argument(s) for the first calculation in the CORDIC_WDATA register. This launches the first calculation.
3. If needed, update the CORDIC_CSR register settings for the next calculation.
4. Program the argument(s) for the next calculation in the CORDIC_WDATA register.
5. Read the result(s) from the CORDIC_RDATA register. This triggers the next calculation.
6. Go to step *3*.

17.3.7 Polling mode

When a new result is available in the CORDIC_RDATA register, the RRDY flag is set in the CORDIC_CSR register. The flag can be polled by reading the register. It is reset by reading the CORDIC_RDATA register (once or twice depending on the NRES field of the CORDIC_CSR register).

Polling the RRDY flag takes slightly longer than reading the CORDIC_RDATA register directly, since the result is not read as soon as it is available. However the processor and bus interface are not stalled while reading the CORDIC_CSR register, so this mode may be of interest if stalling the processor is not acceptable (e.g. if low latency interrupts must be serviced).

17.3.8 Interrupt mode

By setting the interrupt enable (IE) bit in the CORDIC_CSR register, an interrupt is generated whenever the RRDY flag is set. The interrupt is cleared when the flag is reset.

This mode allows the result of the calculation to be read under interrupt service routine, and hence given a priority relative to other tasks. However it is slower than directly reading the result, or polling the flag, due to the interrupt handling delays.

17.3.9 DMA mode

If the DMA write enable (DMAWEN) bit is set in the CORDIC_CSR register, and no operation is pending, a DMA write channel request is generated. The DMA controller can transfer a primary input argument (ARG1) from memory into the CORDIC_WDATA register. Writing into the register deasserts the DMA request. If NARGS = 1 in the CORDIC_CSR register, a second DMA write channel request is generated to transfer the secondary input argument (ARG2) into the CORDIC_WDATA register. When all input arguments have been written, and any ongoing calculation has been completed (by reading the results), a new calculation is started and another DMA write channel request is generated.

If the DMA read enable (DMAREN) bit is set in the CORDIC_CSR register, the RRDY flag going active generates a DMA read channel request. The DMA controller can then transfer the primary result (RES1) from the CORDIC_RDATA register to memory. Reading the register deasserts the DMA request. If NRES = 1 in the CORDIC_CSR register, a second DMA request is generated to read out the secondary result (RES2). When all results have been read, the RRDY flag is deasserted.

The DMA read and write channels can be enabled separately. If both channels are enabled, the CORDIC can autonomously perform repeated calculations on a buffer of data without processor intervention.This allows the processor to perform other tasks. The DMA controller is operating in memory-to-peripheral mode for the write channel, and peripheral-to-memory mode for the read channel. Note that the sequence is started by the processor setting the DMAWEN flag. Thereafter the DMA read and write requests are generated as fast as the CORDIC can process the data.

In some cases, the input data may be stored in memory, and the output is transferred at regular intervals to another peripheral, such as a digital-to-analog converter. In this case, the destination peripheral generates a DMA request each time it needs a new data. The DMA controller can directly fetch the next sample from the CORDIC_RDATA register (in this case the DMA controller is operating in memory-to-peripheral mode, even though the source is a peripheral register). The act of reading the result allows the CORDIC to start a new calculation, which in turn generates a DMA write channel request, and the DMA controller transfers the next input value to the CORDIC_WDATA register. The DMA write channel is enabled (DMAWEN = 1), but the read channel must not be enabled.

In a similar way, data coming from another peripheral, such as an ADC, can be transferred directly to the CORDIC_WDATA register (in peripheral-to-memory mode). The DMA write channel must not be enabled. The CORDIC processes the input data and generate a DMA read request when complete, if DMAREN = 1. The DMA controller then transfers the result from CORDIC_RDATA register to memory (peripheral-to-memory mode).

Note: No DMA request is generated to program the CORDIC_CSR register. DMA mode is therefore only useful when repeatedly performing the same function with the same settings. The scale factor cannot be changed during a series of DMA transfers.

Note: Each DMA request must be acknowledged, as a result of the DMA performing an access to the CORDIC_WDATA or CORDIC_RDATA register. If an extraneous access to the relevant register occurs before this, the acknowledge is asserted prematurely, and could block the DMA channel. Therefore, when the DMA read channel is enabled, CPU access to the CORDIC_RDATA register must be avoided. Similarly, the processor must avoid accessing the CORDIC_WDATA register when the DMA write channel is enabled.

17.4 CORDIC registers

The CORDIC registers can only be accessed in 32-bit word format

17.4.1 CORDIC control/status register (CORDIC_CSR)

Address offset: 0x00

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARG SIZE	RES SIZE	NARG S	NRES	DMA WEN	DMA REN	IEN
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.	Res.	Res.	SCALE[2:0]			PRECISION[3:0]				FUNC[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RRDY**: Result ready flag

0: No new result in output register

1: CORDIC_RDATA register contains new data.

This bit is set by hardware when a CORDIC operation completes. It is reset by hardware when the CORDIC_RDATA register is read (NRES+1) times.

When this bit is set, if the IEN bit is also set, the CORDIC interrupt is asserted. If the DMAREN bit is set, a DMA read channel request is generated. While this bit is set, no new calculation is started.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **ARGSIZE**: Width of input data

0: 32-bit

1: 16-bit

ARGSIZE selects the number of bits used to represent input data.

If 32-bit data is selected, the CORDIC_WDATA register expects arguments in q1.31 format.

If 16-bit data is selected, the CORDIC_WDATA register expects arguments in q1.15 format. The primary argument (ARG1) is written to the least significant half-word, and the secondary argument (ARG2) to the most significant half-word.

Bit 21 **RESSIZE**: Width of output data

0: 32-bit

1: 16-bit

RESSIZE selects the number of bits used to represent output data.

If 32-bit data is selected, the CORDIC_RDATA register contains results in q1.31 format.

If 16-bit data is selected, the least significant half-word of CORDIC_RDATA contains the primary result (RES1) in q1.15 format, and the most significant half-word contains the secondary result (RES2), also in q1.15 format.

Bit 20 **NARGS**: Number of arguments expected by the CORDIC_WDATA register

0: Only one 32-bit write (or two 16-bit values if ARGSIZE = 1) is needed for the next calculation.

1: Two 32-bit values must be written to the CORDIC_WDATA register to trigger the next calculation.

Reads return the current state of the bit.

Bit 19 **NRES**: Number of results in the CORDIC_RDATA register

0: Only one 32-bit value (or two 16-bit values if RESSIZE = 1) is transferred to the CORDIC_RDATA register on completion of the next calculation. One read from CORDIC_RDATA resets the RRDY flag.

1: Two 32-bit values are transferred to the CORDIC_RDATA register on completion of the next calculation. Two reads from CORDIC_RDATA are necessary to reset the RRDY flag.

Reads return the current state of the bit.

Bit 18 **DMAWEN**: Enable DMA write channel

0: Disabled. No DMA write requests are generated.

1: Enabled. Requests are generated on the DMA write channel whenever no operation is pending

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 17 **DMAREN**: Enable DMA read channel

0: Disabled. No DMA read requests are generated.

1: Enabled. Requests are generated on the DMA read channel whenever the RRDY flag is set.

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 16 **IEN**: Enable interrupt.

0: Disabled. No interrupt requests are generated.

1: Enabled. An interrupt request is generated whenever the RRDY flag is set.

This bit is set and cleared by software. A read returns the current state of the bit.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **SCALE[2:0]**: Scaling factor

The value of this field indicates the scaling factor applied to the arguments and/or results. A value n implies that the arguments have been multiplied by a factor 2^{-n}, and/or the results need to be multiplied by 2^{n}. Refer to *Section 17.3.2* for the applicability of the scaling factor for each function and the appropriate range.

Bits 7:4 **PRECISION[3:0]**: Precision required (number of iterations)

0: reserved

1 to 15: (Number of iterations)/4

To determine the number of iterations needed for a given accuracy refer to *Table 115*.

Note that for most functions, the recommended range for this field is 3 to 6.

Bits 3:0 **FUNC[3:0]**: Function

0: Cosine

1: Sine

2: Phase

3: Modulus

4: Arctangent

5: Hyperbolic cosine

6: Hyperbolic sine

7: Arctanh

8: Natural logarithm

9: Square Root

10 to 15: reserved

17.4.2 CORDIC argument register (CORDIC_WDATA)

Address offset: 0x04

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARG[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
ARG[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ARG[31:0]**: Function input arguments

This register is programmed with the input arguments for the function selected in the CORDIC_CSR register FUNC field.

If 32-bit format is selected (CORDIC_CSR.ARGSIZE = 0) and two input arguments are required (CORDIC_CSR.NARGS = 1), two successive writes are required to this register. The first writes the primary argument (ARG1), the second writes the secondary argument (ARG2).

If 32-bit format is selected and only one input argument is required (NARGS = 0), only one write is required to this register, containing the primary argument (ARG1).

If 16-bit format is selected (CORDIC_CSR.ARGSIZE = 1), one write to this register contains both arguments. The primary argument (ARG1) is in the lower half, ARG[15:0], and the secondary argument (ARG2) is in the upper half, ARG[31:16]. In this case, NARGS must be set to 0.

Refer to *Section 17.3.2* for the arguments required by each function, and their permitted range.

When the required number of arguments has been written, the CORDIC evaluates the function designated by CORDIC_CSR.FUNC using the supplied input arguments, provided any previous calculation has completed. If a calculation is ongoing, the ARG1 and ARG 2 values are held pending until the calculation is completed and the results read. During this time, a write to the register cancels the pending operation and overwrite the argument data.

17.4.3 CORDIC result register (CORDIC_RDATA)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES[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
RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RES[31:0]**: Function result

If 32-bit format is selected (CORDIC_CSR.RESSIZE = 0) and two output values are expected (CORDIC_CSR.NRES = 1), this register must be read twice when the RRDY flag is set. The first read fetches the primary result (RES1). The second read fetches the secondary result (RES2) and resets RRDY.

If 32-bit format is selected and only one output value is expected (NRES = 0), only one read of this register is required to fetch the primary result (RES1) and reset the RRDY flag.

If 16-bit format is selected (CORDIC_CSR.RESSIZE = 1), this register contains the primary result (RES1) in the lower half, RES[15:0], and the secondary result (RES2) in the upper half, RES[31:16]. In this case, NRES must be set to 0, and only one read performed.

A read from this register resets the RRDY flag in the CORDIC_CSR register.

17.4.4 CORDIC register map

Table 116. CORDIC register map and reset value

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	CORDIC_CSR	RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARGSIZE	RESSIZE	NARGS	NRES	DMAWEN	DMAREN	IEN	Res.	Res.	Res.	Res.	SCALE [2:0]				PRECISION [3:0]				FUNC [3:0]			
	Reset value	0									0	0	0	0	0	0	0					0	0	0	0	0	1	0	1	0	0	0	0
0x04	CORDIC_WDATA	ARG[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
0x08	CORDIC_RDATA	RES[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 2.2* for the register boundary addresses.

18 Filter math accelerator (FMAC)

18.1 FMAC introduction

The filter math accelerator unit performs arithmetic operations on vectors. It comprises a multiplier/accumulator (MAC) unit, together with address generation logic which allows it to index vector elements held in local memory.

The unit includes support for circular buffers on input and output, which allows digital filters to be implemented. Both finite and infinite impulse response filters can be realized.

The unit allows frequent or lengthy filtering operations to be offloaded from the CPU, freeing up the processor for other tasks. In many cases it can accelerate such calculations compared to a software implementation, resulting in a speed-up of time critical tasks.

18.2 FMAC main features

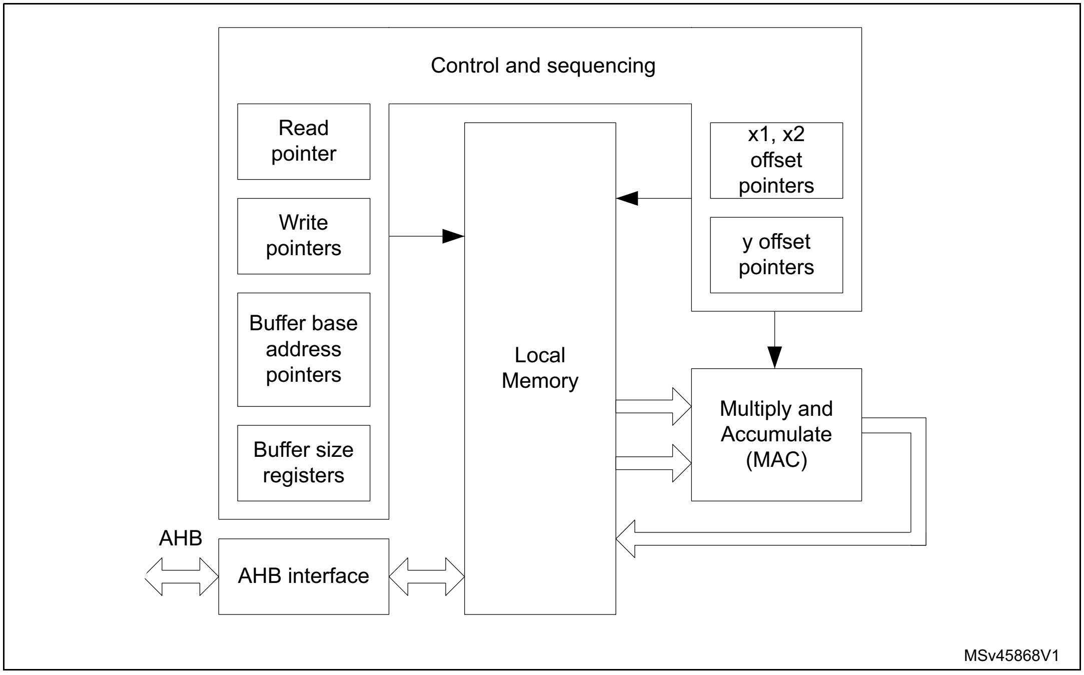
- 16 x 16-bit multiplier
- 24 + 2-bit accumulator with addition and subtraction
- 16-bit input and output data
- 256 x 16-bit local memory
- Up to three areas can be defined in memory for data buffers (two input, one output), defined by programmable base address pointers and associated size registers
- Input and output buffers can be circular
- Filter functions: FIR, IIR (direct form 1)
- Vector functions: Dot product, convolution, correlation
- AHB slave interface
- DMA read and write data channels

18.3 FMAC functional description

18.3.1 General description

The FMAC is shown in *Figure 41*.

Figure 41. Block diagram



The unit is built around a fixed point multiplier and accumulator (MAC). The MAC can take two 16-bit input signed values from memory, multiply them together and add them to the contents of the accumulator. The address of the input values in memory is determined using a set of pointers. These pointers can be loaded, incremented, decremented or reset by the internal hardware. The pointer and MAC operations are controlled by a built-in sequencer in order to execute the requested operation.

To calculate a dot product, the two input vectors are loaded into the local memory by the processor or DMA controller, and the requested operation is selected and started. Each pair of input vector elements is fetched from memory, multiplied together and accumulated. When all the vector elements have been processed, the contents of the accumulator are stored in the local memory, from where they can be read out by the processor or DMA.

The finite impulse response (FIR) filter operation (also known as convolution) consists in repeatedly calculating the dot product of the coefficient vector and a vector of input samples, the latter being shifted by one sample delay, with the least recent sample being discarded and a new sample added, at each repetition.

The infinite impulse response (IIR) filter operation is the convolution of the feedback coefficients with the previous output samples, added to the result of the FIR convolution.

A more detailed description of the filter operations is given in *Section 18.3.6: Filter functions*.

18.3.2 Local memory and buffers

The unit contains a 256 x 16-bit read/write memory which is used for local storage:

- Input values (the elements of the input vectors) are stored in two buffers, X1 and X2.
- Output values (the results of the operations) are stored in another buffer, Y.
- The locations and sizes of the buffers are designated as follows:
  - x1_base: the base address of the X1 buffer
  - x2_base: the base address of the X2 buffer
  - y_base: the base address of the Y buffer
  - x1_buf_size: the number of 16-bit addresses allocated to the X1 buffer
  - x2_buf_size: the number of 16-bit addresses allocated to the X2 buffer
  - y_buf_size: the number of 16-bit addresses allocated to the Y buffer.

These parameters are programmed in the corresponding registers when configuring the unit.

The CPU (or DMA controller) can initialize the contents of each buffer using the Initialization functions (*Section 18.3.5: Initialization functions*) and writing to the write data register. The data is transferred to the location within the target buffer indicated by a write pointer. After each new write, the write pointer is incremented. When the write pointer reaches the end of the allocated buffer space, it wraps back to the base address. This feature is used to load the elements of a vector prior to an operation, or to initialize a filter and load filter coefficients.

Buffer configuration

The buffer sizes and base address offsets must be configured in the X1, X2 and Y buffer configuration registers. For each function, the required buffer size is specified in the function description in *Section 18.3.6: Filter functions*. The base addresses can be chosen anywhere in internal memory, provided that all buffers fit within the internal memory address range (0x00 to 0xFF), that is, base address + buffer size must be less than 256.

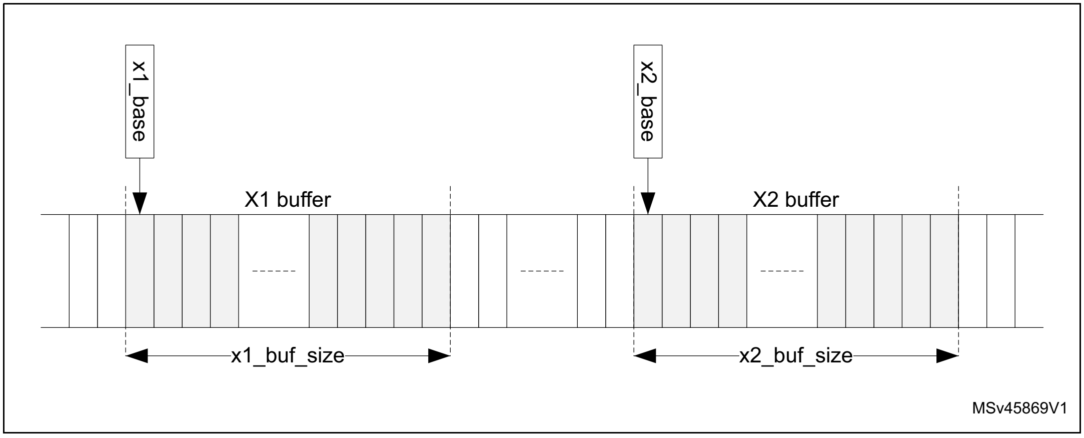
There is no constraint on the size and location of the buffers (they can overlap or even coincide exactly). For filter functions it is recommended not to overlap buffers as this can lead to erroneous behavior.

When circular buffer operation is required, an optional “headroom”, d, can be added to the buffer size. Furthermore, a watermark level can be set, to regulate the CPU or DMA activity. The value of d and the watermark level should be chosen according to the application performance requirements. For maximum throughput, the input buffer should never go empty, so d should be somewhat greater than the watermark level, allowing for any interrupt or DMA latency. On the other hand, if the input data can not be provided as fast as the unit can process them, the buffer can be allowed to empty waiting for the next data to be written, so d can be equal to the watermark level (to ensure that no overflow occurs on the input).

18.3.3 Input buffers

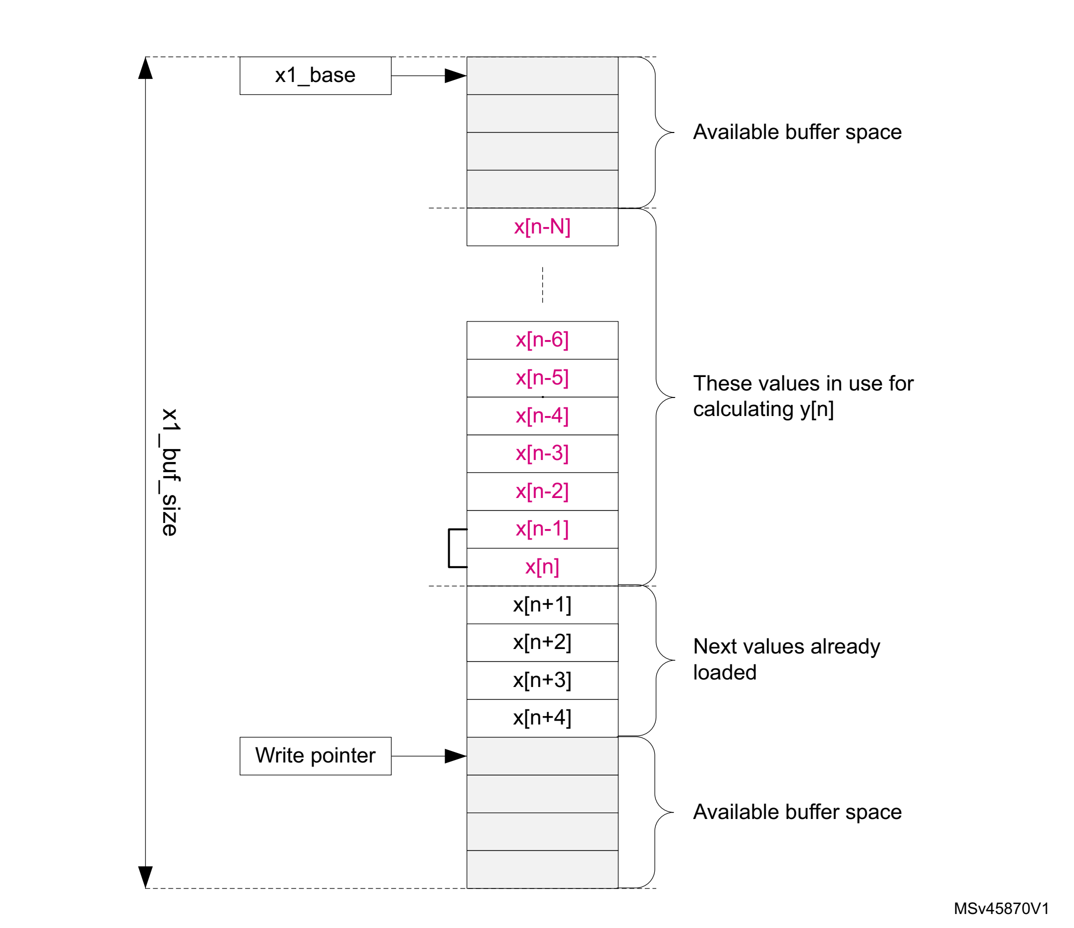
The X1 and X2 buffers are used to store data for input to the MAC. Each multiplication takes a value from the X1 buffer and a value from the X2 buffer and multiplies them together. A pointer in the control unit generates the read address offset (relative to the buffer base address) for each value. The pointers are managed by hardware according to the current function.

Figure 42. Input buffer areas



The X1 buffer can be used as a circular buffer, in which case new data are continually transferred into the input buffer whenever space is available. Pre-loading this buffer is optional for digital filters, since if no input samples have been written in the buffer when the operation is started, it is flagged as empty, which triggers the CPU or DMA to load new samples until there are enough to begin operation. Pre-loading is nevertheless useful in the case of a vector operation, that is, the input data is already available in system memory and circular operation is not required.

Figure 43. Circular input buffer



The X2 buffer can only be used in vector mode (that is not circular), and needs to be pre-loaded, except if the contents of the buffer do not change from one operation to the next. For filter functions, the X2 buffer is used to store the filter coefficients.

When operating as a circular buffer, the space allocated to the buffer (x1_buf_size) should generally be bigger than the number of elements in use for the current calculation, so that there are always new values available in the buffer. *Figure 43* illustrates the layout of the buffer for a filter operation. While calculating an output sample y[n], the unit uses a set of N+1 input samples, x[n-N] to x[n]. When this is finished, the unit starts the calculation of y[n+1], using the set of input samples x[n-N+1] to x[n+1]. The least-recent input sample, x[n-N], drops out of the input set, and a new sample, x[n+1], is added to it.

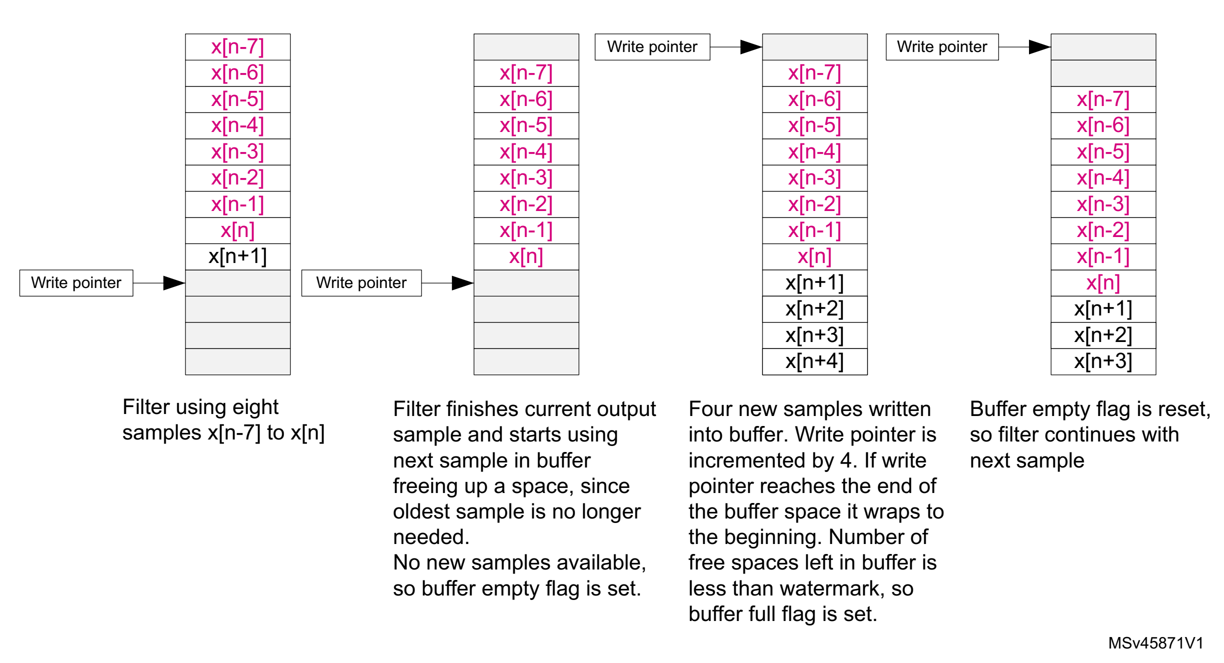
The processor, or DMA controller, must ensure that the new sample x[n+1] is available in the buffer space when required. If not, the buffer is flagged as empty, which stalls the execution of the unit until a new sample is added. No underflow condition is signaled on the X1 buffer.

Note: If the flow of samples is controlled by a timer or other peripheral such as an ADC, the buffer regularly goes empty, since the filter processes each new sample faster than the source can provide it. This is an essential feature of filter operation.

If the number of free spaces in the buffer is less than the watermark threshold programmed in the FULL_WM bitfield of the FMAC_X1BUFCFG register, the buffer is flagged as full. As long as the full flag is not set, interrupts are generated, if enabled, to request more data for the buffer. The watermark allows several data to be transferred under one interrupt, without danger of overflow. Nevertheless, if an overflow does occur, the OVFL error flag is set and the write data is ignored. The write pointer is not incremented in the event of an overflow.

The operation of the X1 buffer during a filtering operation is illustrated in *Figure 44*. This example shows an 8-tap FIR filter with a watermark set to four.

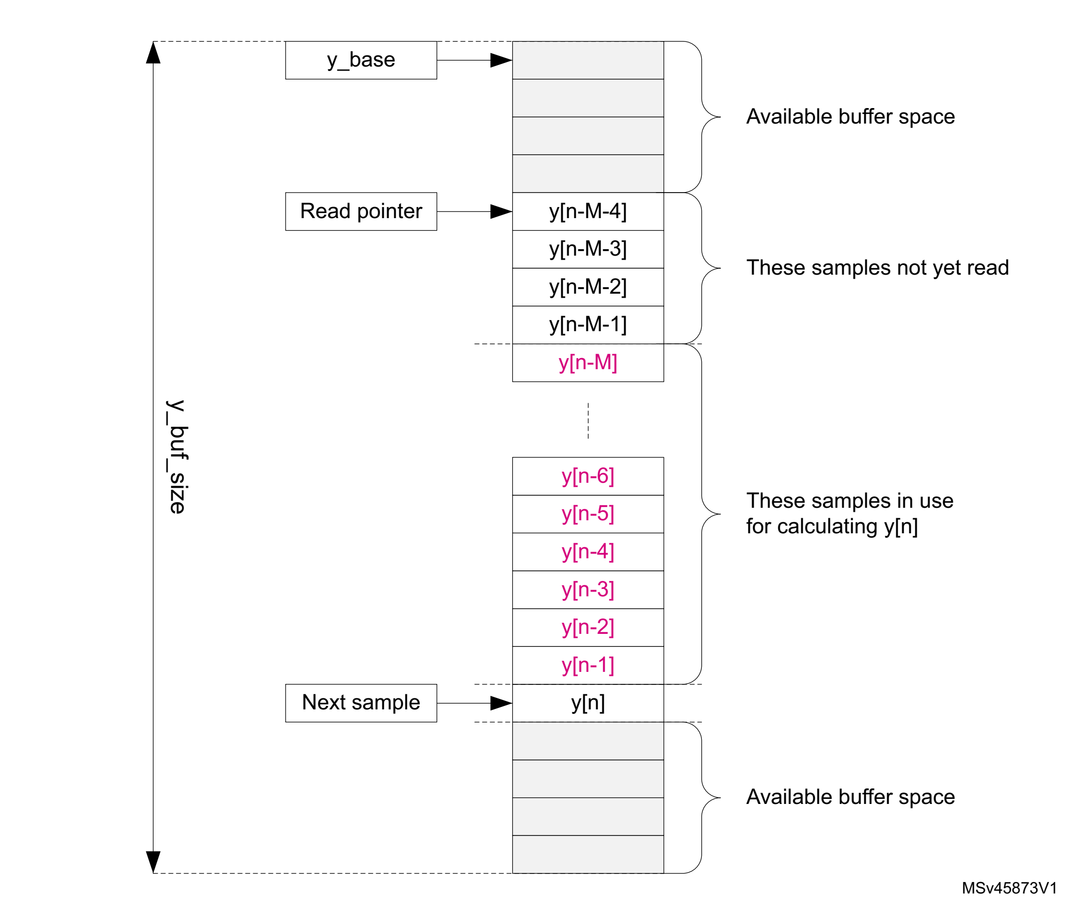
Figure 44. Circular input buffer operation


x[n-7]
x[n-6]
x[n-5]
x[n-4]
x[n-3]
x[n-2]
x[n-1]
x[n]
x[n+1]
x[n+2]
x[n+3]
x[n+4]
Write pointer
Filter using eight samples x[n-7] to x[n]
Filter finishes current output sample and starts using next sample in buffer freeing up a space, since oldest sample is no longer needed.
No new samples available, so buffer empty flag is set.
Four new samples written into buffer. Write pointer is incremented by 4. If write pointer reaches the end of the buffer space it wraps to the beginning. Number of free spaces left in buffer is less than watermark, so buffer full flag is set.
Buffer empty flag is reset, so filter continues with next sample
MSv45871V1

18.3.4 Output buffer

The Y (output) buffer is used to store the output of an accumulation. Each new output value is stored in the buffer until it is read by the processor or DMA controller. Each time a read access is made to the read data register, the read data is fetched from the address indicated by the read pointer. This pointer is incremented after each read, and wraps back to the base address when it reaches the end of the allocated Y buffer space.

Figure 45. Circular output buffer



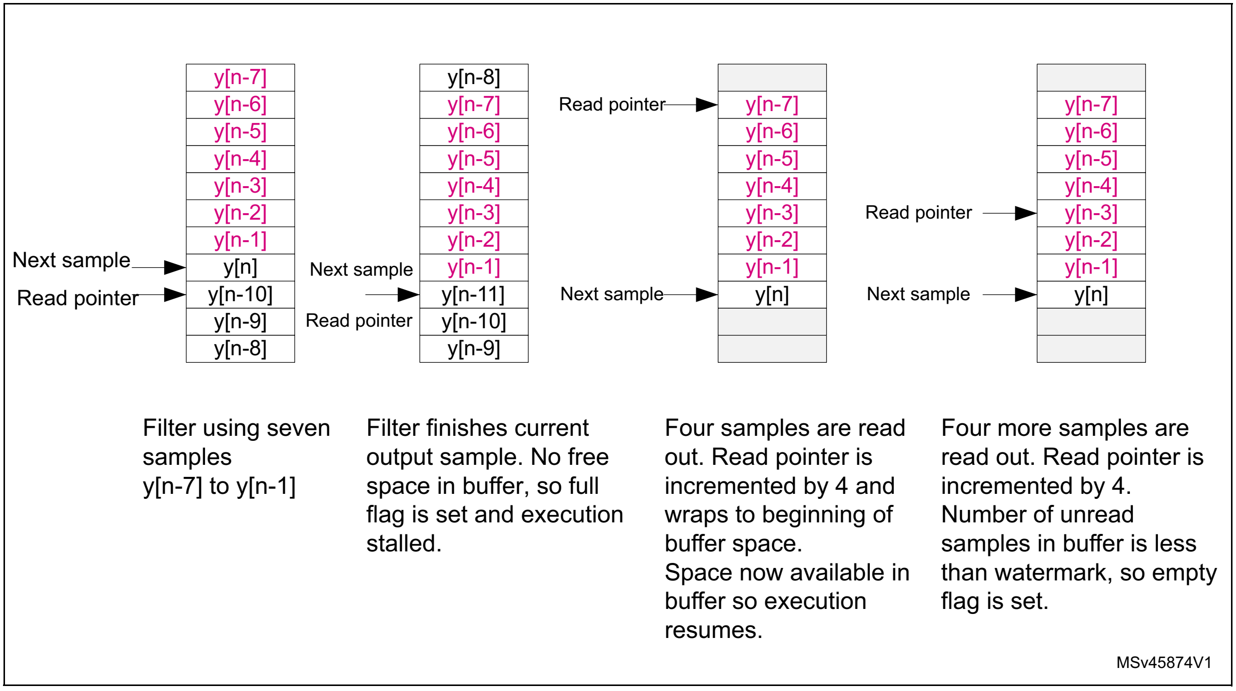
The Y buffer can also operate as a circular buffer. If the address for the next output value is the same as that indicated by the read pointer (ie. an unread sample), then the buffer is flagged as full and execution stalled until the sample is read.

In the case of IIR filters, the Y buffer is used to store the set of M previous output samples, y[n-M] to y[n-1], used for calculating the next output sample y[n]. Each time a new sample is added to the set, the least recent sample y[n-M] drops out.

If the number of unread data in the buffer is less than the watermark threshold programmed in the EMPTY_WM bitfield of the FMAC_YBUFCFG register, the buffer is flagged as empty. As long as the empty flag is not set, interrupts or DMA requests are generated, if enabled, to request reads from the buffer. The watermark allows several data to be transferred under one interrupt, without danger of underflow. Nevertheless, if an underflow does occur, the UNFL error flag is set. In this case, the read pointer is not incremented and the read operation returns the content of the memory at the read pointer address.

The operation of the Y buffer in circular mode is illustrated in *Figure 46*. This example shows a 7-tap IIR filter with a watermark set to four.

Figure 46. Circular output buffer operation



Filter using seven samples y[n-7] to y[n-1]

Filter finishes current output sample. No free space in buffer, so full flag is set and execution stalled.

Four samples are read out. Read pointer is incremented by 4 and wraps to beginning of buffer space. Space now available in buffer so execution resumes.

Four more samples are read out. Read pointer is incremented by 4. Number of unread samples in buffer is less than watermark, so empty flag is set.

18.3.5 Initialization functions

The following functions initialize the FMAC unit. They are triggered by writing the appropriate value in the FUNC bitfield of the FMAC_PARAM register, with the START bit set. The P and Q bitfields must also contain the appropriate parameter values for each function as detailed below. The R bitfield is not used. When the function completes, the START bit is automatically reset by hardware.

During initialization, it is recommended that the DMA requests and interrupts be disabled. The transfer of data into the FMAC memory can be done by software or by memory-to-memory DMA transfers, since no flow control is required.

Load X1 buffer

This function pre-loads the X1 buffer with N values, starting from the address in X1_BASE. Successive writes to the FMAC_WDATA register load the write data into the X1 buffer and increment the write address. The write pointer points to the address X1_BASE + N when the function completes.

The function can be used to pre-load the buffer with the elements of a vector, or to initialize the input storage elements of a filter.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X1 buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

Load X2 buffer

This function pre-loads the X2 buffer with N + M values, starting from the address in X2_BASE. Successive writes to the FMAC_WDATA register load the write data into the X2 buffer and increment the write address.

The function can be used to pre-load the buffer with the elements of a vector, or the coefficients of a filter. In the case of an IIR, the N feed-forward and M feed-back coefficients are concatenated and loaded together into the X2 buffer. The total number of coefficients is equal to N + M. For an FIR, there are no feedback coefficients, so M = 0.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X2 buffer starting from address X2_BASE.
- The parameter Q contains the number of values, M, to be loaded into the X2 buffer starting from address X2_BASE + N.
- The parameter R is not used.

The function completes when N + M writes have been performed to the FMAC_WDATA register.

Load Y buffer

This function pre-loads the Y buffer with N values, starting from the address in Y_BASE. Successive writes to the FMAC_WDATA register load the write data into the Y buffer and increment the write address. The read pointer points to the address Y_BASE + N when the function completes.

The function can be used to pre-load the feedback storage elements of an IIR filter.

Parameters

- The parameter P contains the number of values to be loaded into the Y buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

18.3.6 Filter functions

The following filter functions are supported by the FMAC unit. These functions are triggered by writing the corresponding value in the FUNC bitfield of the FMAC_PARAM register with the START bit set. The P, Q and R bitfields must also contain the appropriate parameter values for each function as detailed below. The filter functions continue to run until the START bit is reset by software.

Convolution (FIR filter)

$\underline{\mathbf{Y}} = \underline{\mathbf{B}}*\underline{\mathbf{X}}$

$$y_n = 2^R \cdot \sum_{k=0}^{N} b_k x_{n-k}$$

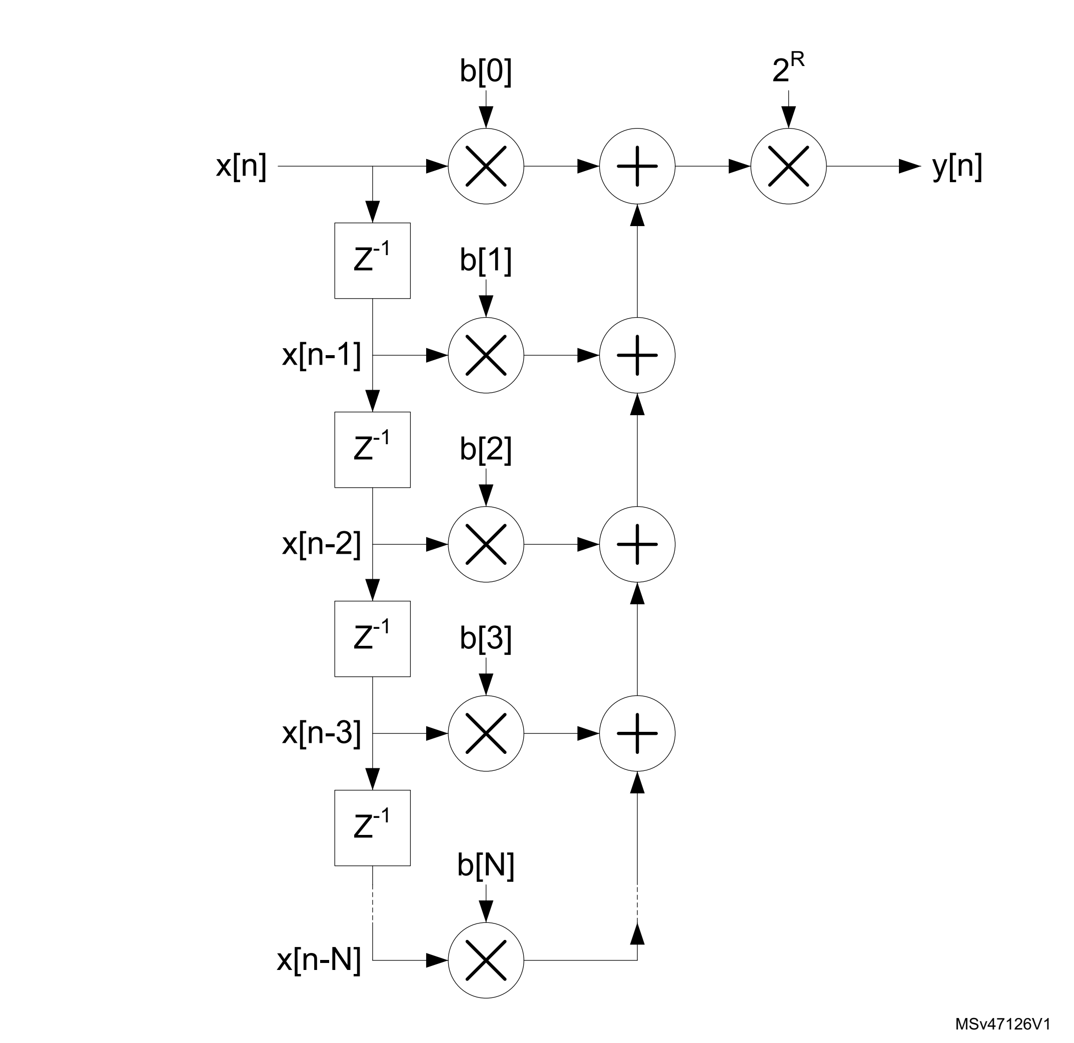
This function performs a convolution of a vector **B** of length N+1 and a vector **X** of indefinite length. The elements of **Y** for incrementing values of n are calculated as the dot product,

y_n = $\underline{\mathbf{B}}.\underline{\mathbf{X}}_{\mathbf{n}}$, where $\underline{\mathbf{X}}_{\mathbf{n}}$ = $[x_{n-N},...,x_n]$ is composed of the N+1 elements of $\underline{\mathbf{X}}$ at indexes n - N to n.

This function corresponds to a finite impulse response (FIR) filter, where vector $\underline{\mathbf{B}}$ contains the filter coefficients and vector $\underline{\mathbf{X}}$ the sampled data.

The structure of the filter (direct form) is shown in *Figure 47*.

Figure 47. FIR filter structure



Note that the cross correlation vector can be calculated by reversing the order of the coefficient vector $\underline{\mathbf{B}}$.

Input:

- X1 buffer contains the elements of vector $\underline{\mathbf{X}}$. It is a circular buffer of length N + 1 + d.
- X2 buffer contains the elements of vector $\underline{\mathbf{B}}$. It is a fixed buffer of length N + 1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length d.

Parameters:

- The parameter P contains the length, N+1, of the coefficient vector **B** in the range [2:127].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7]
- The parameter Q is not used.

The function completes when the START bit in the FMAC_PARAM register is reset by software.

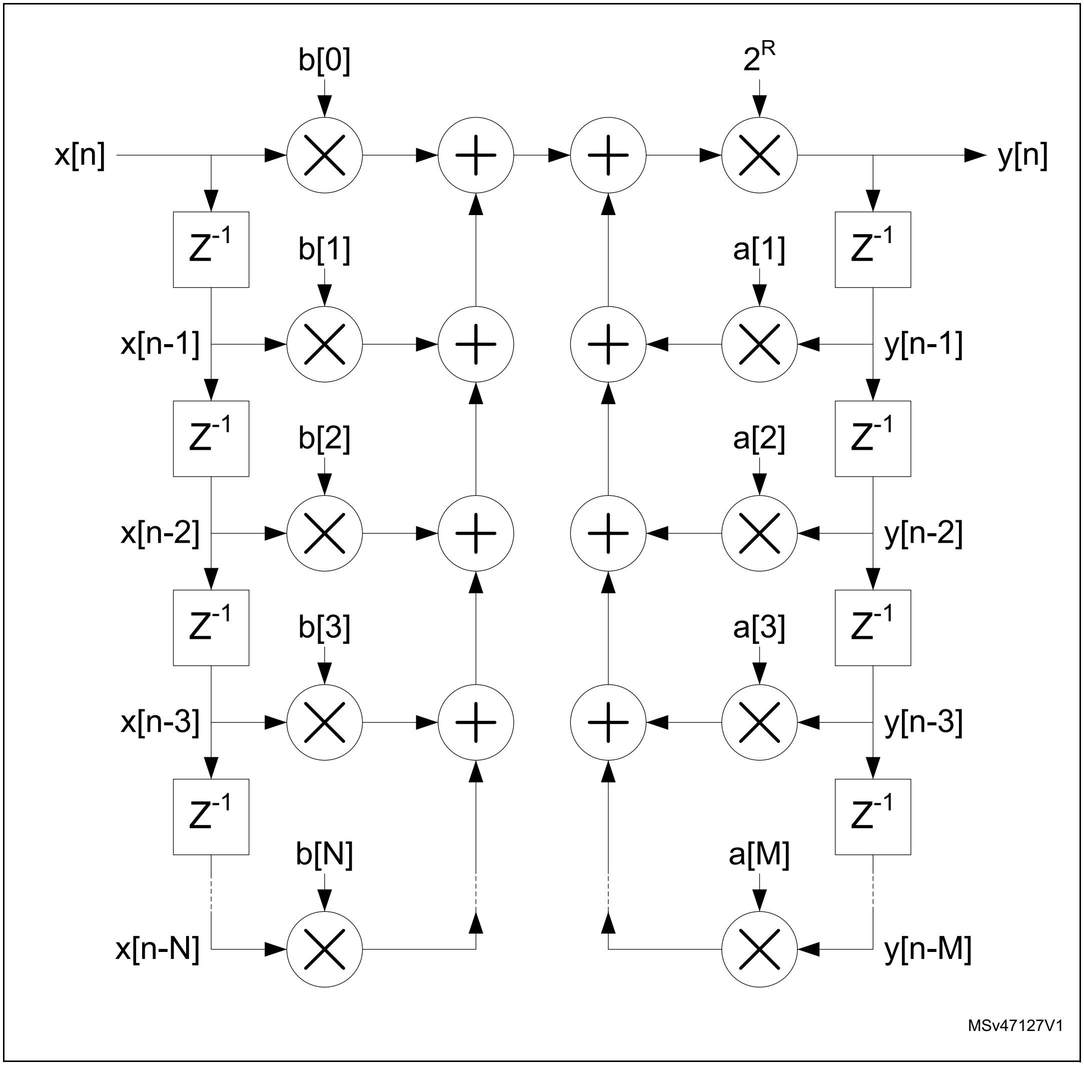
IIR filter

$\underline{\mathbf{Y}} = \underline{\mathbf{B}} * \underline{\mathbf{X}} + \underline{\mathbf{A}} * \underline{\mathbf{Y}}'$

$$y_n = 2^R \cdot \left(\sum_{k=0}^{N} b_k x_{n-k} + \sum_{k=1}^{M} a_k y_{n-k} \right)$$

This function implements an infinite impulse response (IIR) filter. The filter output vector **Y** is the convolution of a coefficient vector **B** of length N+1 and a vector **X** of indefinite length, plus the convolution of the delayed output vector **Y’** with a second coefficient vector **A**, of length M. The elements of **Y** for incrementing values of n are calculated as $y_n = \underline{\mathbf{B}}.\underline{\mathbf{X}}_{\mathbf{n}} + \underline{\mathbf{A}}.\underline{\mathbf{Y}}_{\mathbf{n-1}}$, where $\underline{\mathbf{X}}_{\mathbf{n}} = [x_{n-N},\ldots,x_n]$ comprises the N+1 elements of **X** at indexes n - N to n, while $\underline{\mathbf{Y}}_{\mathbf{n-1}} = [y_{n-M},\ldots,y_{n-1}]$ comprises the M elements of **Y** at indexes n - M to n - 1. The structure of the filter (direct form 1) is shown in *Figure 48*.

Figure 48. IIR filter structure (direct form 1)



Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1+ d.
- X2 buffer contains the elements of coefficient vectors **B** and **A** concatenated (b_0, b_1, b_2..., b_N, a_1, a_2, ..., a_M). It is a fixed buffer of length M+N+1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length M + d.

Parameters

- The parameter P contains the length, N + 1, of the coefficient vector **B** in the range [2:64].
- The parameter Q contains the length, M, of the coefficient vector **A** in the range [1:63].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7].

The function completes when the START bit in the FMAC_PARAM register is reset by software.

18.3.7 Fixed point representation

The FMAC operates in fixed point signed integer format. Input and output values are q1.15.

In q1.15 format, numbers are represented by one sign bit and 15 fractional bits (binary decimal places). The numeric range is therefore -1 (0x8000) to 1 - 2^{-15} (0x7FFF).

The accumulator has 26 bits, of which 22 are fractional and 4 are integer/sign (q4.22). This allows it to support partial accumulation sums in the range -8 (0x2000000) to +7.99999976 (0x1FFFFFF). A programmable gain from 0dB to 42dB in steps of 6dB can be applied at the output of the accumulator.

Note that the content of the accumulator is not saturated if the numeric range is exceeded. Partial sums whose value is greater than +7.99999976 or less than -8, wrap but this is harmless provided subsequent accumulations undo the wrapping. Nevertheless, the SAT flag in the FMAC_SR register is set if wrapping occurs, and generates an interrupt if the SATIEN bit is set in the FMAC_CR register. This helps in debugging the filter.

The data output by the accumulator can optionally be saturated, after application of the programmable gain, by setting the CLIPEN bit in the FMAC_CR register. If this bit is set, then any value which exceeds the numeric range of the q1.15 output, is set to 1 - 2^{-15} or -1, according to the sign. If clipping is not enabled, the unused accumulator bits after applying the gain is simply truncated.

18.3.8 Implementing FIR filters with the FMAC

The FMAC supports FIR filters of length N, where N is the number of taps or coefficients. The minimum local memory requirement for a FIR filter of length N is 2N + 1:

- N coefficients
- N input samples
- 1 output sample

Since the local memory size is 256, the maximum value for N is 127.

If maximum throughput is required, it may be necessary to allocate a small amount of extra space, d1 and d2, to the input and output sample buffers respectively, to ensure that the filter never stalls waiting for a new input sample, or waiting for the output sample to be read. In this case, the local memory requirement is 2N + d1 + d2.

The buffers should be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N;
- Y_BUF_SIZE = d2 (or 1 if no extra space is required)

The buffer base addresses can be allocated anywhere, but the X2 buffer must not overlap with the others, or else the coefficients are overwritten. An example configuration could be:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = 2N + d1

However, if the memory space is limited, the X1 and Y buffer areas can be overlapped, such that each output sample takes the place of the oldest input sample, which is no longer required:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = N

In this case, Y_BUF_SIZE = X1_BUF_SIZE = N + d1, so that the buffers remain in sync.

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $log_2(d1)$*, otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to* $log_2(d2)$*.*

The filter coefficients **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. There is no point in pre-loading the Y buffer, since for the FIR filter there is no feedback path.

After configuring and initializing the buffers, the FMAC_CR register should be programmed according to the method used for writing and reading data to and from the FMAC memory.

Three methods are supported:

- Polling: No DMA request or Interrupt request is generated. Software must check that the X1_FULL flag is low before writing to WDATA, or that the Y_EMPTY flag is low before reading from RDATA.
- Interrupt: The interrupt request is asserted while the X1_FULL flag is low, for writes, or when the Y_EMPTY flag is low, for reads.
- DMA: DMA requests are asserted on the DMA write channel while the X1_FULL flag is low, and on the read channel while the Y_EMPTY flag is low.

Different methods can be used for read and for write. However it is not recommended to use both interrupts and DMA requests for the same operation[a]. The valid combinations are listed in *Table 117*.

Table 117. Valid combinations for read and write methods

WIEN	RIEN	DMAWEN	DMAREN	Write	Read
0	0	0	0	Polling	Polling
0	1	0	0	Polling	Interrupt
1	0	0	0	Interrupt	Polling
1	1	0	0	Interrupt	Interrupt
0	0	0	1	Polling	DMA
0	0	1	0	DMA	Polling
0	0	1	1	DMA	DMA
0	1	1	0	DMA	Interrupt
1	0	0	1	Interrupt	DMA

[a] If both interrupts and DMA requests are enabled then only DMA should perform the transfer.

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 8 (FIR filter);
- P = N (number of coefficients);
- Q = “Don’t care”;
- R = Gain;
- START = 1;

If less than N + d - 2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine should check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples).

When 2^{EMPTY_WM} output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine should check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high.

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

18.3.9 Implementing IIR filters with the FMAC

The FMAC supports IIR filters of length N, where N is the number of feed-forward taps or coefficients. The number of feedback coefficients, M, can be any value from 1 to N-1. Only direct form 1 implementations can be realized, so filters designed for other forms need to be converted.

The minimum memory requirement for an IIR filter with N feed-forward coefficients and M feed-back coefficients is 2N + 2M:

- N + M coefficients
- N input samples
- M output samples

If M = N-1, then the maximum filter length that can be implemented is N = 64.

As for the FIR, for maximum throughput a small amount of additional space, d1 and d2, should be allowed in the input and output buffer size respectively, making the total memory requirement 2M + 2N + d1 + d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N + M;
- Y_BUF_SIZE = M + d2;

The buffer base addresses can be allocated anywhere, but must not overlap. An example configuration is given below:

- X2_BASE = 0;
- X1_BASE = N + M;
- Y_BASE = 2N + M + d1;

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $\log_2$(d1), otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to $\log_2$(d2).

The filter coefficients (N feed-forward followed by M feedback) **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. The Y buffer can optionally be pre-loaded with any number of values up to a maximum of M. This has the effect of initializing the feedback delay line.

After configuring the buffers, the FMAC_CR register should be programmed in the same way as for the FIR filter (see *Section 18.3.8: Implementing FIR filters with the FMAC*).

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 9 (IIR filter);
- P = N (number of feed-forward coefficients);
- Q = M (number of feed-back coefficients);
- R = Gain;
- START = 1;

If less than N + d - $2^{\text{FULL\_WM}}$ values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write $2^{\text{FULL\_WM}}$ additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine should check the X1FULL flag after every $2^{\text{FULL\_WM}}$ writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

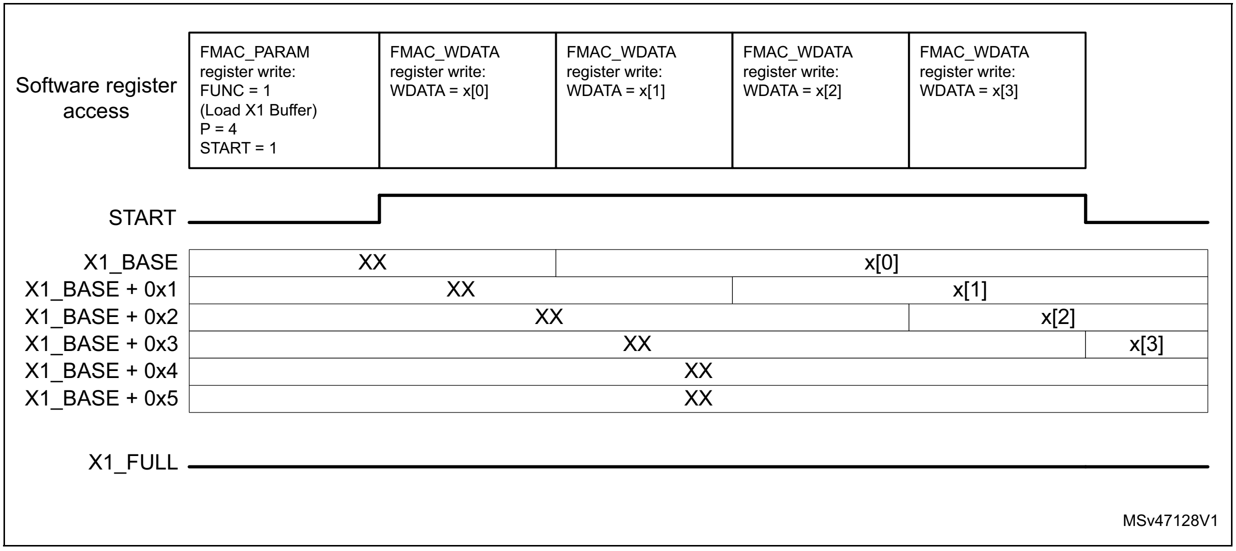
The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples). The first sample is calculated using the first N samples in the X1 buffer, and the first M samples in the Y buffer (whether or not they are preloaded. The first output sample is written into the Y buffer at Y_BASE + M.

When $2^{\text{EMPTY\_WM}}$ new output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read $2^{\text{EMPTY\_WM}}$ samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine should check the YEMPTY flag after every $2^{\text{EMPTY\_WM}}$ reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

18.3.10 Examples of filter initialization

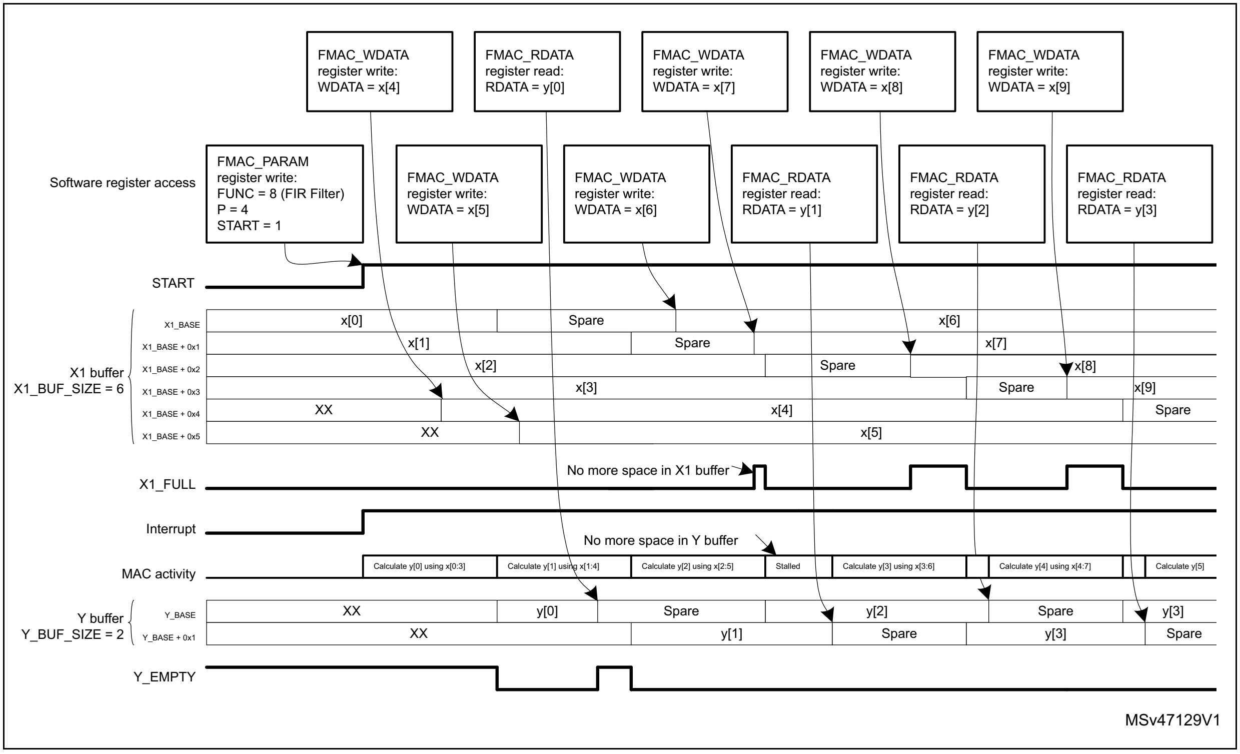
Figure 49. X1 buffer initialization


Software register access
FMAC_PARAM register write: FUNC = 1 (Load X1 Buffer) P = 4 START = 1
FMAC_WDATA register write: WDATA = x[0]
FMAC_WDATA register write: WDATA = x[1]
FMAC_WDATA register write: WDATA = x[2]
FMAC_WDATA register write: WDATA = x[3]
START
X1_BASE
X1_BASE + 0x1
X1_BASE + 0x2
X1_BASE + 0x3
X1_BASE + 0x4
X1_BASE + 0x5
XX
x[0]
x[1]
x[2]
x[3]
X1_FULL
MSv47128V1

The example in *Figure 49* illustrates an X1 buffer pre-load with four samples (P = 4). The buffer size is six (X1_BUF_SIZE = 6). The initialization is launched by programming the FMAC_PARAM register with the START bit set. The four samples are then written to FMAC_WDATA, and transferred into local memory from X1_BASE onwards. The START bit resets after the fourth sample has been written. At this point, the X1 buffer contains the four samples, in order of writing, and the write pointer (next empty space) is at X1_BASE + 0x4.

18.3.11 Examples of filter operation

Figure 50. Filtering example 1


FMAC_WDATA register write: WDATA = x[4]
FMAC_RDATA register read: RDATA = y[0]
FMAC_WDATA register write: WDATA = x[7]
FMAC_WDATA register write: WDATA = x[8]
FMAC_WDATA register write: WDATA = x[9]
Software register access
FMAC_PARAM register write: FUNC = 8 (FIR Filter) P = 4 START = 1
FMAC_WDATA register write: WDATA = x[5]
FMAC_WDATA register write: WDATA = x[6]
FMAC_RDATA register read: RDATA = y[1]
FMAC_RDATA register read: RDATA = y[2]
FMAC_RDATA register read: RDATA = y[3]
START
X1 buffer
X1_BUF_SIZE = 6
X1_BASE
X1_BASE + 0x1
X1_BASE + 0x2
X1_BASE + 0x3
X1_BASE + 0x4
X1_BASE + 0x5
x[0]
Spare
x[6]
x[1]
Spare
x[7]
x[2]
Spare
x[8]
x[3]
Spare
x[9]
XX
x[4]
Spare
XX
x[5]
X1_FULL
No more space in X1 buffer
Interrupt
No more space in Y buffer
MAC activity
Calculate y[0] using x[0:3]
Calculate y[1] using x[1:4]
Calculate y[2] using x[2:5]
Stalled
Calculate y[3] using x[3:6]
Calculate y[4] using x[4:7]
Calculate y[5]
Y buffer
Y_BUF_SIZE = 2
Y_BASE
Y_BASE + 0x1
XX
y[0]
Spare
y[2]
Spare
y[3]
XX
y[1]
Spare
y[3]
Spare
Y_EMPTY
MSv47129V1

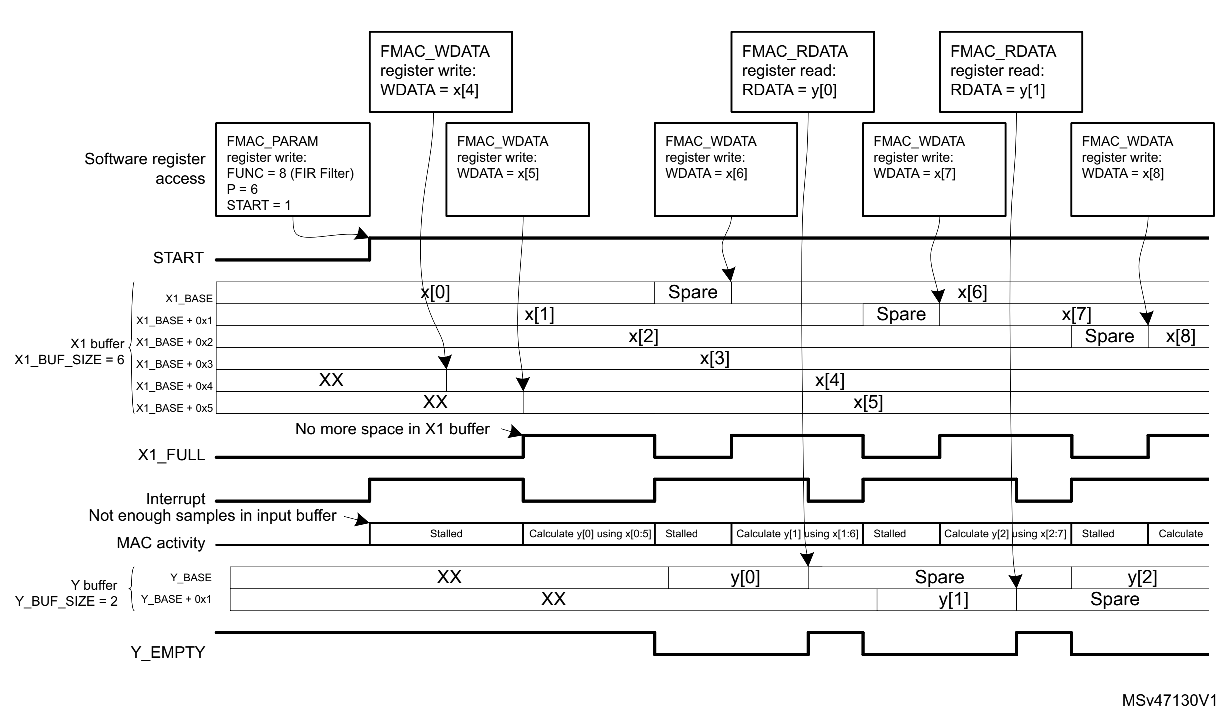
The example in *Figure 50* illustrates the beginning of a filter operation. The filter has four taps (P=4). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 49*. So the filter starts calculating the first output sample, y[0], immediately after the START bit is set. Since the X1FULL flag is not set (due to two uninitialized spaces in the X1 buffer), the interrupt is asserted straight away, to request new data. The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer.

In the mean time, the FMAC finishes calculating the first output sample, y[0], and writes it into the Y buffer, causing the Y_EMPTY flag to go low. At the same time, the x[0] sample is discarded, as it is no longer required, freeing up its location in memory (at X1_BASE). The FMAC can immediately start work on the second output sample, y[1], since all the required input samples x[1:5] are present in the X1 buffer.

Since the Y_EMPTY flag is low, the interrupt remains active after the processor finishes writing x[5]. The processor reads y[0] from the FMAC_RDATA register, freeing up its location in the Y buffer. There are now no samples in the output buffer since y[1] is still being calculated, so the Y_EMPTY flag goes high. Nevertheless, the interrupt remains active, because there is still free space in the X1 buffer, which the processor next fills with x[6], and so on.

Note: In this example, the processor can fill the input buffer more quickly than the FMAC can process them, so the X1_full flag regularly goes active. However, it struggles to read the Y buffer fast enough, so the FMAC stalls regularly waiting for space to be freed up in the Y buffer. This means the filter is not executing at maximum throughput. The reason is that the *filter length is small and the processor relatively slow, in this example. So increasing the Y buffer size would not help.*

Figure 51. Filtering example 2


FMAC_WDATA register write: WDATA = x[4]
FMAC_RDATA register read: RDATA = y[0]
FMAC_RDATA register read: RDATA = y[1]
Software register access
FMAC_PARAM register write: FUNC = 8 (FIR Filter) P = 6 START = 1
FMAC_WDATA register write: WDATA = x[5]
FMAC_WDATA register write: WDATA = x[6]
FMAC_WDATA register write: WDATA = x[7]
FMAC_WDATA register write: WDATA = x[8]
START
X1 buffer
X1_BUF_SIZE = 6
X1_BASE
X1_BASE + 0x1
X1_BASE + 0x2
X1_BASE + 0x3
X1_BASE + 0x4
X1_BASE + 0x5
x[0]
Spare
x[6]
x[1]
Spare
x[7]
x[2]
Spare
x[8]
x[3]
XX
x[4]
XX
x[5]
No more space in X1 buffer
X1_FULL
Interrupt
Not enough samples in input buffer
MAC activity
Stalled
Calculate y[0] using x[0:5]
Stalled
Calculate y[1] using x[1:6]
Stalled
Calculate y[2] using x[2:7]
Stalled
Calculate
Y buffer
Y_BUF_SIZE = 2
Y_BASE
Y_BASE + 0x1
XX
y[0]
Spare
y[2]
XX
y[1]
Spare
Y_EMPTY
MSv47130V1

The example in *Figure 51* illustrates the beginning of the same filter operation, but this time the filter has six taps (P=6). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 49*. Because there are not enough samples in the input buffer, the X1FULL flag is not set, so the interrupt is asserted straight away, to request new data. The FMAC is stalled.

The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer. As soon as there are six unused samples in the X1 buffer, the X1_FULL flag goes active (since the buffer size is six), causing the interrupt to go inactive. The FMAC starts calculating the first output sample, y[0]. Since this requires all six input samples, there are no free spaces in the X1 buffer and so the X1_FULL flag remains active. Only when the FMAC finishes calculating y[0] and writes it into the Y buffer, can x[0] be discarded, freeing up a space in the X1 buffer, and deasserting X1_FULL. At the same time, the Y_EMPTY flag goes inactive. Both these flag states cause the interrupt to be asserted, requesting the processor to write a new input sample, first of all, and then read the output sample just calculated. The FMAC remains stalled until a new input sample is written.

In this example, the processor has to wait for the FMAC to finish calculating the current output sample, before it can write a new input sample, and therefore the X1 buffer regularly goes empty, stalling the FMAC. This can be avoided by allowing some extra space in the input buffer.

18.3.12 Filter design tips

The FMAC architecture imposes some constraints detailed below, on the design of digital filters.

1. Implementation of direct form 2, or transposed forms, is not efficient. Filters which have been designed for such forms should be converted to direct form 1.
2. Cascaded filters must either be combined into a single stage, or implemented as separate filters. In the latter case, multiple sets of filter coefficients can be pre-loaded into the memory, one set per stage, and only the X2_BASE address changed to select which set is used. The most efficient method of implementing a multi-stage filter is to pre-load a large X1 buffer with input samples, run the IIR filter function on it using the first stage coefficients, and store the output samples back in memory. Then change the X2_BASE pointer to point to the 2nd stage coefficients, and reload the input buffer with the output of the first stage (with a gain if required), before running the IIR function again. The procedure is repeated for all stages. Once the final stage samples have been transferred back into system memory, the input buffer can be loaded with the next set of input samples, and a new round of calculations started. Note that the N sample input buffer of each stage must be pre-loaded first of all with the N-1 last inputs from the previous round, plus one new sample, in order to keep continuity between each round. Similarly, the output buffer of each stage must be loaded with the last M samples from the previous round, for the same reason.
3. The use of direct form 1 for IIR designs can lead to large positive or negative partial sums in the accumulator, if for example a large step occurs on the input, or some of the filter coefficients' absolute values are >1. Since the accumulator is limited to 26 bits, the biggest value that it can handle without wrapping (changing sign) is 0x1FFFFFF positive or 0x2000000 negative. This corresponds to 3.99999988 and -4 respectively in q3.23 fixed point format. Wrapping does not represent a problem provided the wrapping is “undone” before the end of the accumulation. However this is not always the case when a filter is starting up and can lead to unexpected results. Consider pre-loading the output buffer with suitable values to avoid this.
4. The IIR filter has feed-forward (numerator) coefficients [b_0, b_1, ..., b_{N-1}], and feed-back (denominator) coefficients [1, a_1, ..., a_M]. Many IIR filters require some of the denominator coefficients to have an absolute value greater than 1 to achieve a steep roll-off in the frequency response. Given that the coefficients are coded in fixed point q1.15 format, this is not possible. Nevertheless, by scaling the denominator coefficients by a factor 2^{-R}, such that 2^{-R}.[1, a_1, ..., a_M] are all less than 1, such filters can be implemented. However an inverse gain of 2^R must be applied at the output of the accumulator to compensate the scaling. This has an adverse effect on the signal-to-noise ratio.

18.4 FMAC registers

18.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG)

Address offset: 0x00

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FULL_WM[1:0]		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
X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
rw	rw	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 25:24 **FULL_WM[1:0]**: Watermark for buffer full flag

Defines the threshold for setting the X1 buffer full flag when operating in circular mode. The flag is set if the number of free spaces in the buffer is less than 2^{FULL_WM}.

0: Threshold = 1

1: Threshold = 2

2: Threshold = 4

3: Threshold = 8

Setting a threshold greater than 1 allows several data to be transferred into the buffer under one interrupt.

Threshold should be set to 1 if DMA write requests are enabled (DMAWEN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **X1_BUF_SIZE[7:0]**: Allocated size of X1 buffer in 16-bit words

The minimum buffer size is the number of feed-forward taps in the filter (+ the watermark threshold - 1).

Bits 7:0 **X1_BASE[7:0]**: Base address of X1 buffer

18.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG)

Address offset: 0x04

Reset value: 0x0000 0000

Access: word 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
X2_BUF_SIZE[7:0]								X2_BASE[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 **X2_BUF_SIZE[7:0]**: Size of X2 buffer in 16-bit words

This bitfield can not be modified when a function is ongoing (START = 1).

Bits 7:0 **X2_BASE[7:0]**: Base address of X2 buffer

The X2 buffer base address can be modified while START=1, for example to change coefficient values. The filter should be stalled when doing this, since changing the coefficients while a calculation is ongoing affects the result.

18.4.3 FMAC Y buffer configuration register (FMAC_YBUFCFG)

Address offset: 0x08

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	EMPTY_WM[1:0]		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
Y_BUF_SIZE[7:0]								Y_BASE[7:0]							
rw	rw	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 25:24 **EMPTY_WM[1:0]**: Watermark for buffer empty flag

Defines the threshold for setting the Y buffer empty flag when operating in circular mode. The flag is set if the number of unread values in the buffer is less than 2^{EMPTY_WM}.

0: Threshold = 1
1: Threshold = 2
2: Threshold = 4
3: Threshold = 8

Setting a threshold greater than 1 allows several data to be transferred from the buffer under one interrupt.

Threshold should be set to 1 if DMA read requests are enabled (DMAREN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **Y_BUF_SIZE[7:0]**: Size of Y buffer in 16-bit words

For FIR filters, the minimum buffer size is 1 (+ the watermark threshold). For IIR filters the minimum buffer size is the number of feedback taps (+ the watermark threshold).

Bits 7:0 **Y_BASE[7:0]**: Base address of Y buffer

18.4.4 FMAC parameter register (FMAC_PARAM)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
START	FUNC[6:0]							R[7:0]							
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
Q[7:0]								P[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **START**: Enable execution

0: Stop execution

1: Start execution

Setting this bit triggers the execution of the function selected in the FUNC bitfield. Resetting it by software stops any ongoing function. For initialization functions, this bit is reset by hardware.

Bits 30:24 **FUNC[6:0]**: Function

0: Reserved

1: Load X1 buffer

2: Load X2 buffer

3: Load Y buffer

4 to 7: Reserved

8: Convolution (FIR filter)

9: IIR filter (direct form 1)

10 to 127: Reserved

This bitfield can not be modified when a function is ongoing (START = 1)

Bits 23:16 **R[7:0]**: Input parameter R.

The value of this parameter is dependent on the function.

This bitfield can not be modified when a function is ongoing (START = 1)

Bits 15:8 **Q[7:0]**: Input parameter Q.

The value of this parameter is dependent on the function.

This bitfield can not be modified when a function is ongoing (START = 1)

Bits 7:0 **P[7:0]**: Input parameter P.

The value of this parameter is dependent on the function

This bitfield can not be modified when a function is ongoing (START = 1)

18.4.5 FMAC control register (FMAC_CR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: word 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.	RESET
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLIP EN	Res.	Res.	Res.	Res.	Res.	DMA WEN	DMA REN	Res.	Res.	Res.	SAT IEN	UNFL IEN	OVFL IEN	WIEN	RIEN
rw						rw	rw				rw	rw	rw	rw	rw

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

Bit 16 **RESET**: Reset FMAC unit

This resets the write and read pointers, the internal control logic, the FMAC_SR register and the FMAC_PARAM register, including the START bit if active. Other register settings are not affected. This bit is reset by hardware.

0: Reset inactive

1: Reset active

Bit 15 **CLIPEN**: Enable clipping

0: Clipping disabled. Values at the output of the accumulator which exceed the q1.15 range, wrap.

1: Clipping enabled. Values at the output of the accumulator which exceed the q1.15 range are saturated to the maximum positive or negative value (+1 or -1) according to the sign.

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **DMAWEN**: Enable DMA write channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the X1 buffer is not full.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bit 8 **DMAREN**: Enable DMA read channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the Y buffer is not empty.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SATIEN**: Enable saturation error interrupts

0: Disabled. No interrupts are generated upon saturation detection.

1: Enabled. An interrupt request is generated if the SAT flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 3 **UNFLIEN**: Enable underflow error interrupts

0: Disabled. No interrupts are generated upon underflow detection.

1: Enabled. An interrupt request is generated if the UNFL flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 2 **OVFLIEN**: Enable overflow error interrupts

0: Disabled. No interrupts are generated upon overflow detection.
1: Enabled. An interrupt request is generated if the OVFL flag is set
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 1 **WIEN**: Enable write interrupt

0: Disabled. No write interrupt requests are generated.
1: Enabled. An interrupt request is generated while the X1 buffer FULL flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 0 **RIEN**: Enable read interrupt

0: Disabled. No read interrupt requests are generated.
1: Enabled. An interrupt request is generated while the Y buffer EMPTY flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

18.4.6 FMAC status register (FMAC_SR)

Address offset: 0x14

Reset value: 0x0000 0001

Access: word 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.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1 FULL	Y EMPTY
					r	r	r							r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SAT**: Saturation error flag

Saturation occurs when the result of an accumulation exceeds the numeric range of the accumulator.
0: No saturation detected
1: Saturation detected. If the SATIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 9 **UNFL**: Underflow error flag

An underflow occurs when a read is made from FMAC_RDATA when no valid data is available in the Y buffer.
0: No underflow detected
1: Underflow detected. If the UNFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 8 **OVFL**: Overflow error flag

An overflow occurs when a write is made to FMAC_WDATA when no free space is available in the X1 buffer.
0: No overflow detected
1: Overflow detected. If the OVFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

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

Bit 1 **X1FULL**: X1 buffer full flag

The buffer is flagged as full if the number of available spaces is less than the FULL_WM threshold. The number of available spaces is the difference between the write pointer and the least recent sample currently in use.

0: X1 buffer not full. If the WIEN bit is set, the interrupt request is asserted until the flag is set. If DMAWEN is set, DMA write channel requests are generated until the flag is set.

1: X1 buffer full.

This flag is set and cleared by hardware, or by a reset.

Note: after the last available space in the X1 buffer is filled there is a delay of 3 clock cycles before the X1FULL flag goes high. To avoid any risk of overflow it is recommended to insert a software delay after writing to the X1 buffer before reading the FMAC_SR. Alternatively, a FULL_WM threshold of 2 can be used.

Bit 0 **YEMPTY**: Y buffer empty flag

The buffer is flagged as empty if the number of unread data is less than the EMPTY_WM threshold. The number of unread data is the difference between the read pointer and the current output destination address.

0: Y buffer not empty. If the RIEN bit is set, the interrupt request is asserted until the flag is set. If DMAREN is set, DMA read channel requests are generated until the flag is set.

1: Y buffer empty.

This flag is set and cleared by hardware, or by a reset.

Note: after the last sample is read from the Y buffer there is a delay of 3 clock cycles before the YEMPTY flag goes high. To avoid any risk of underflow it is recommended to insert a software delay after reading from the Y buffer before reading the FMAC_SR. Alternatively, an EMPTY_WM threshold of 2 can be used.

18.4.7 FMAC write data register (FMAC_WDATA)

Address offset: 0x18

Reset value: 0x0000 0000

Access: word and half-word 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
WDATA[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 **WDATA[15:0]**: Write data

When a write access to this register occurs, the write data are transferred to the address offset indicated by the write pointer. The pointer address is automatically incremented after each write access.

18.4.8 FMAC read data register (FMAC_RDATA)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: word and half-word 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
RDATA[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 **RDATA[15:0]**: Read data

When a read access to this register occurs, the read data are the contents of the Y output buffer at the address offset indicated by the READ pointer. The pointer address is automatically incremented after each read access.

18.4.9 FMAC register map

Table 118.FMAC 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	FMAC_X1BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	FULL_WM [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	FMAC_X2BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X2_BUF_SIZE[7:0]								X2_BASE[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	FMAC_YBUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY_WM [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Y_BUF_SIZE[7:0]								Y_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	FMAC_PARAM	START	FUNC[6:0]							R[7:0]								Q[7:0]								P[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	0	0
0x10	FMAC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET	CLIPEN	Res.	Res.	Res.	Res.	Res.	DMAWEN	DMAREN	Res.	Res.	Res.	SATIEN	UNFLIEN	OVFLIEN	WIEN	RIEN
	Reset value																0	0						0	0				0	0	0	0	0
0x14	FMAC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1FULL	YEMPTY
	Reset value																						0	0	0							0	1
0x18	FMAC_WDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	FMAC_RDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2* for the register boundary addresses.

19 Flexible static memory controller (FSMC)

19.1 Introduction

The flexible static memory controller (FSMC) includes two memory controllers:

- The NOR/PSRAM memory controller
- The NAND memory controller

This memory controller is also named flexible memory controller (FMC).

19.2 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, and NAND Flash memory. Its main purposes are:

- to translate AHB transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique chip select. The FMC performs only one access at a time to an external device.

The main features of the FMC controller are the following:

- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR Flash memory/OneNAND Flash memory
  - PSRAM (4 memory banks)
  - Ferroelectric RAM (FRAM)
  - NAND Flash memory with ECC hardware to check up to 8 Kbytes of data
- Interface with parallel LCD modules, supporting Intel 8080 and Motorola 6800 modes.
- Burst mode support for faster access to synchronous devices such as NOR Flash memory, PSRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16-bit wide data bus
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Write enable and byte lane select outputs for use with PSRAM, SRAM devices
- External asynchronous wait control
- Write FIFO with 16 x32-bit depth

The Write FIFO is common to all memory controllers and consists of:

- a Write Data FIFO which stores the AHB data to be written to the memory (up to 32 bits) plus one bit for the AHB transfer (burst or not sequential mode)
- a Write Address FIFO which stores the AHB address (up to 28 bits) plus the AHB data size (up to 2 bits). When operating in burst mode, only the start address is stored except when crossing a page boundary (for PSRAM). In this case, the AHB burst is broken into two FIFO entries.

At startup the FMC pins must be configured by the user application. The FMC I/O pins which are not used by the application can be used for other purposes.

The FMC registers that define the external device type and associated characteristics are usually set at boot time and do not change until the next reset or power-up. However, the settings can be changed at any time.

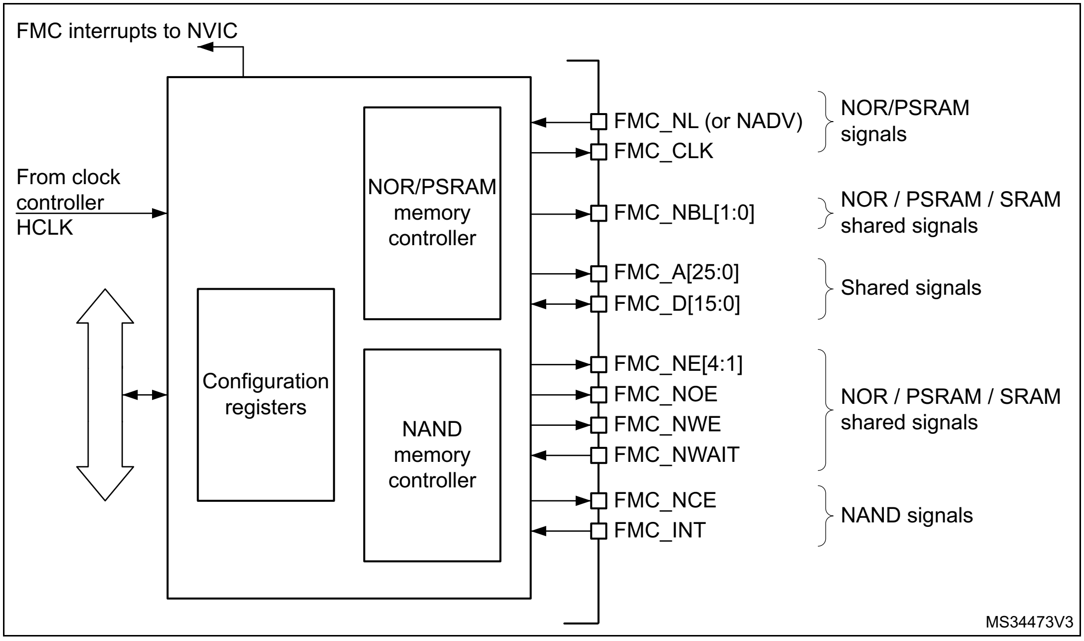
19.3 FMC block diagram

The FMC consists of the following main blocks:

- The AHB interface (including the FMC configuration registers)
- The NOR Flash/PSRAM/SRAM controller

The block diagram is shown in the figure below.

Figure 52. FMC block diagram



19.4 AHB interface

The AHB slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AHB transactions are translated into the external device protocol. In particular, if the selected external memory is 16- or 8-bit wide, 32-bit wide transactions on the AHB are split into consecutive 16- or 8-bit accesses. The FMC chip select (FMC_NEx) does not toggle between the consecutive accesses except in case of Access mode D when the Extended mode is enabled.

The FMC generates an AHB error in the following conditions:

- When reading or writing to an FMC bank (Bank 1 to 4) which is not enabled.
- When reading or writing to the NOR Flash bank while the FACCEN bit is reset in the FMC_BCRx register.

The effect of an AHB error depends on the AHB master which has attempted the R/W access:

- If the access has been attempted by the Cortex®-M4 with FPU CPU, a hard fault interrupt is generated.
- If the access has been performed by a DMA controller, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

The AHB clock (HCLK) is the reference clock for the FMC.

19.4.1 Supported memories and transactions

General transaction rules

The requested AHB transaction data size can be 8-, 16- or 32-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- AHB transaction size and memory data size are equal

  There is no issue in this case.
- AHB transaction size is greater than the memory size:

  In this case, the FMC splits the AHB transaction into smaller consecutive memory accesses to meet the external data width. The FMC chip select (FMC_NEx) does not toggle between the consecutive accesses. If the bus turnaround timings is configured to any other value than 0, the FMC chip select (FMC_NEx) toggles between the consecutive accesses. This feature is required when interfacing with FRAM memory.

- AHB transaction size is smaller than the memory size:

  The transfer may or not be consistent depending on the type of external device:

  – Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM)

    In this case, the FMC allows read/write transactions and accesses the right data through its byte lanes NBL[1:0].

    Bytes to be written are addressed by NBL[1:0].

    All memory bytes are read (NBL[1:0] are driven low during read transaction) and the useless ones are discarded.

  – Accesses to devices that do not have the byte select feature (NOR and NAND Flash memories)

    This situation occurs when a byte access is requested to a 16-bit wide Flash memory. Since the device cannot be accessed in Byte mode (only 16-bit words can be read/written from/to the Flash memory), Write transactions and Read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

Wrap support for NOR Flash/PSRAM

Wrap burst mode for synchronous memories is not supported. The memories must be configured in Linear burst mode of undefined length.

Configuration registers

The FMC can be configured through a set of registers. Refer to *Section 19.6.6*, for a detailed description of the NOR Flash/PSRAM controller registers. Refer to *Section 19.7.7*, for a detailed description of the NAND Flash registers.

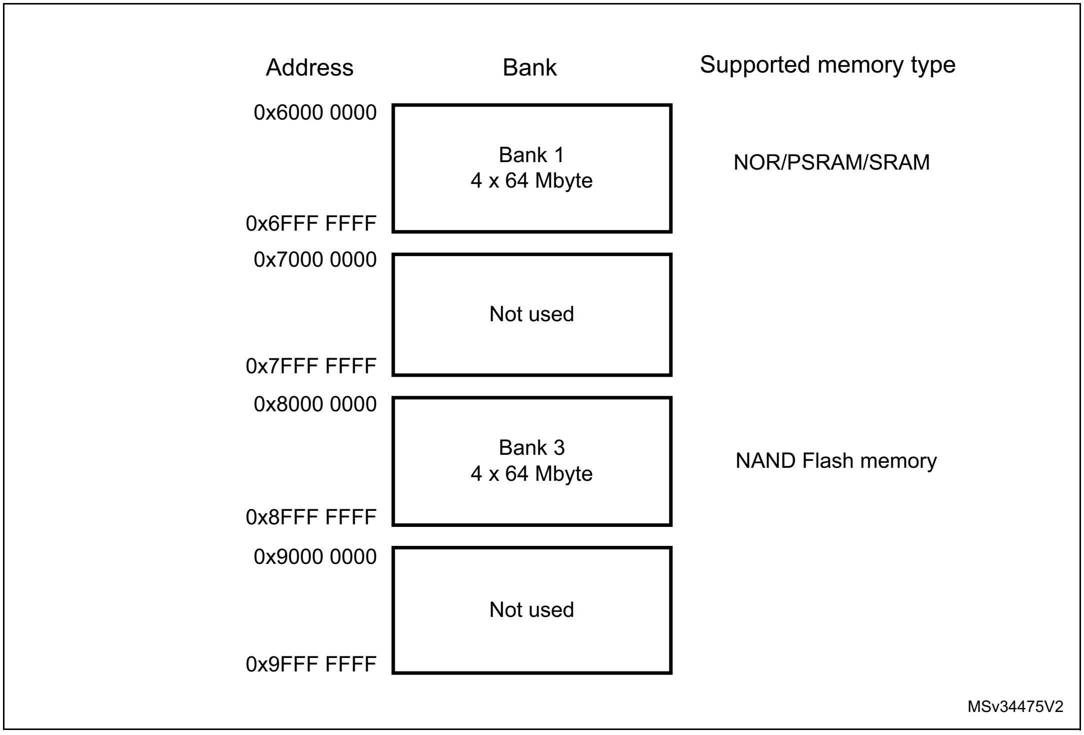
19.5 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see *Figure 53*):

- Bank 1 used to address up to 4 NOR Flash memory or PSRAM devices. This bank is split into 4 NOR/PSRAM subbanks with 4 dedicated chip selects, as follows:
  – Bank 1 - NOR/PSRAM 1
  – Bank 1 - NOR/PSRAM 2
  – Bank 1 - NOR/PSRAM 3
  – Bank 1 - NOR/PSRAM 4
- Bank 3 used to address NAND Flash memory devices.The MPU memory attribute for this space must be reconfigured by software to Device.

For each bank the type of memory to be used can be configured by the user application through the Configuration register.

Figure 53. FMC memory banks



19.5.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in *Table 119*.

Table 119. NOR/PSRAM bank selection

HADDR[27:26][1]	Selected bank
00	Bank 1 - NOR/PSRAM 1
01	Bank 1 - NOR/PSRAM 2
10	Bank 1 - NOR/PSRAM 3
11	Bank 1 - NOR/PSRAM 4

[1] HADDR are internal AHB address lines that are translated to external memory.

The HADDR[25:0] bits contain the external memory address. Since HADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 120. NOR/PSRAM External memory address

Memory width[1]	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	HADDR[25:0]	64 Mbytes x 8 = 512 Mbit
16-bit	HADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbit

[1] In case of a 16-bit external memory width, the FMC internally uses HADDR[25:1] to generate the address for external memory FMC_A[24:0].
Whatever the external memory width, FMC_A[0] should be connected to external memory address A[0].

19.5.2 NAND Flash memory address mapping

The NAND bank is divided into memory areas as indicated in *Table 121*.

Table 121. NAND memory mapping and timing registers

Start address	End address	FMC bank	Memory space	Timing register
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND Flash	Attribute	FMC_PATT (0x8C)
0x8000 0000	0x83FF FFFF		Common	FMC_PMEM (0x88)

For NAND Flash memory, the common and attribute memory spaces are subdivided into three sections (see in *Table 122* below) located in the lower 256 Kbytes:

- Data section (first 64 Kbytes in the common/attribute memory space)
- Command section (second 64 Kbytes in the common / attribute memory space)
- Address section (next 128 Kbytes in the common / attribute memory space)

Table 122. NAND bank selection

Section name	HADDR[17:16]	Address range
Address section	1X	0x020000-0x03FFFF
Command section	01	0x010000-0x01FFFF
Data section	00	0x000000-0x0FFFF

The application software uses the 3 sections to access the NAND Flash memory:

- **To sending a command to NAND Flash memory**, the software must write the command value to any memory location in the command section.
- **To specify the NAND Flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.
- **To read or write data,** the software reads or writes the data from/to any memory location in the data section.

Since the NAND Flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

19.6 NOR Flash/PSRAM controller

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM, FRAM and ROM
  - 8 bits
  - 16 bits
- PSRAM (CellularRAM™)
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed
- NOR Flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The FMC outputs a unique chip select signal, NE[4:1], per bank. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC Clock (FMC_CLK) is a submultiple of the HCLK clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, Bank 1 must be configured in Synchronous mode (see *Section 19.6.6: NOR/PSRAM controller registers*). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AHB data size has to be the same as the memory data width (MWID) otherwise the FMC_CLK frequency is changed depending on AHB data transaction (refer to *Section 19.6.5: Synchronous transactions* for FMC_CLK divider ratio formula).

The size of each bank is fixed and equal to 64 Mbytes. Each bank is configured through dedicated registers (see *Section 19.6.6: NOR/PSRAM controller registers*).

The programmable memory parameters include access times (see *Table 123*) and support for wait management (for PSRAM and NOR Flash accessed in Burst mode).

Table 123. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	AHB clock cycle (HCLK)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	AHB clock cycle (HCLK)	1	15
NBL setup	Duration of the byte lanes setup phase	Asynchronous	AHB clock cycle (HCLK)	0	3
Data setup	Duration of the data setup phase	Asynchronous	AHB clock cycle (HCLK)	1	256
Data hold	Duration of the data hold phase	Asynchronous	AHB clock cycle (HCLK)	0	3
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read / write	AHB clock cycle (HCLK)	0	15
Clock divide ratio	Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)	Synchronous	AHB clock cycle (HCLK)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (CLK)	2	17

19.6.1 External memory interface signals

Table 124, *Table 125* and *Table 126* list the signals that are typically used to interface with NOR Flash memory, SRAM and PSRAM.

Note: The prefix “N” identifies the signals that are active low.

NOR Flash memory, non-multiplexed I/Os

Table 124. Non-multiplexed I/O NOR Flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FMC

The maximum capacity is 512 Mbits (26 address lines).

NOR Flash memory, 16-bit multiplexed I/Os

Table 125. 16-bit multiplexed I/O NOR Flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FMC

The maximum capacity is 512 Mbits.

PSRAM/FRAM/SRAM, non-multiplexed I/Os

Table 126. Non-multiplexed I/Os PSRAM/SRAM

FMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (CellularRAM™ i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os

Table 127. 16-Bit multiplexed I/O PSRAM

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)

Table 127. 16-Bit multiplexed I/O PSRAM (continued)

FMC signal name	I/O	Function
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (CellularRAM™ i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)

The maximum capacity is 512 Mbits (26 address lines).

19.6.2 Supported memories and transactions

Table 128 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR Flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

Table 128. NOR Flash/PSRAM: example of supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NOR Flash (muxed I/Os and nonmuxed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	N	-
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32	16	Y	-

Table 128. NOR Flash/PSRAM: example of supported memories and transactions (continued)

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
PSRAM (multiplexed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32	16	Y	-
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32	16	Y	-
SRAM and ROM	Asynchronous	R	8 / 16	16	Y	-
	Asynchronous	W	8 / 16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses Use of byte lanes NBL[1:0]

19.6.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the internal clock (HCLK)
- In Synchronous mode (read or write), all output signals change on the rising edge of HCLK. Whatever the CLKDIV value, all outputs change as follows:
  – NOEL/NWEL/ NEL/NADVL/ NADVH /NBLL/ Address valid outputs change on the falling edge of FMC_CLK clock.
  – NOEH/ NWEH / NEH/ NOEH/NBLH/ Address invalid outputs change on the rising edge of FMC_CLK clock.

19.6.4 NOR Flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR Flash, PSRAM, SRAM, FRAM)

- Signals are synchronized by the internal clock HCLK. This clock is not issued to the memory
- The FMC always samples the data before de-asserting the NOE signal. This guarantees that the memory data hold timing constraint is met (minimum Chip Enable high to data transition is usually 0 ns)
- If the Extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B.
- If the Extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in mode 1 or mode 2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP = 0x10 in the FMC_BCRx register).

Mode 1 - SRAM/FRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

Figure 54. Mode 1 read access waveforms

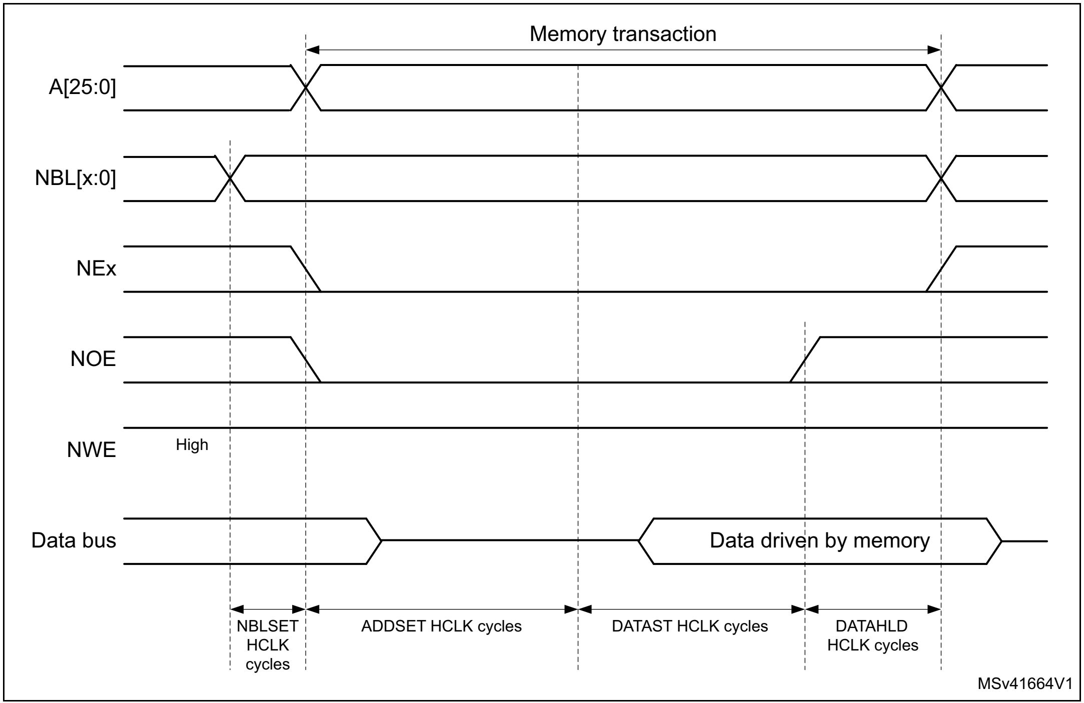
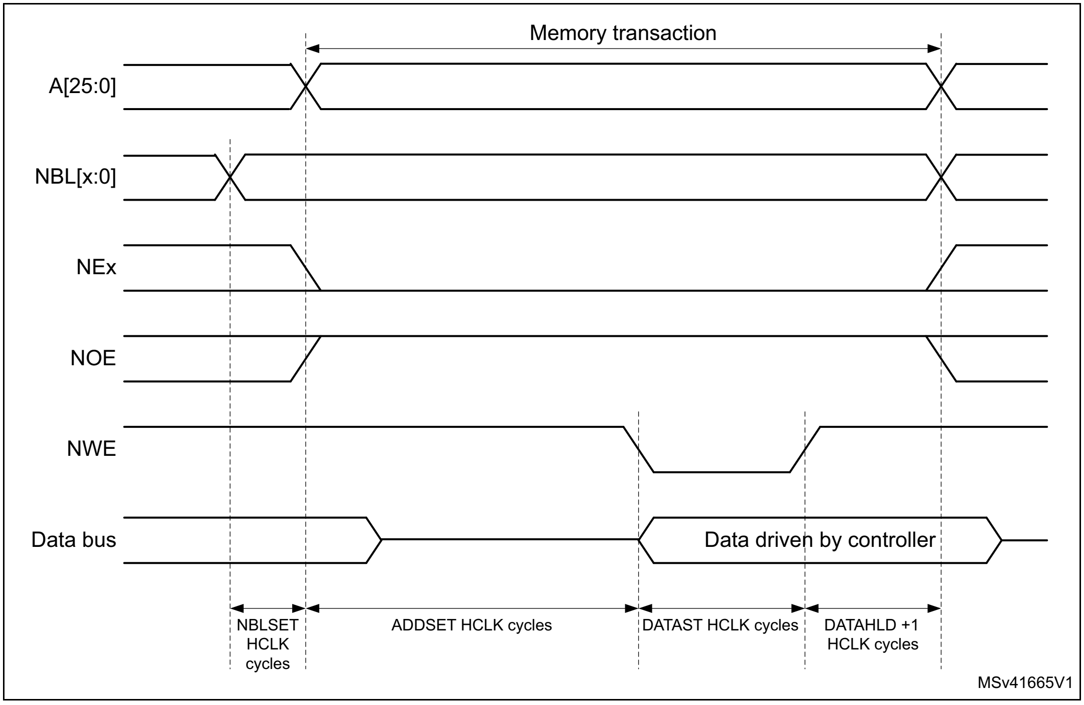


Figure 55. Mode 1 write access waveforms



The DATAHLD time at the end of the read and write transactions guarantee the address and data hold time after the NOE/NWE rising edge. The DATAST value must be greater than zero (DATAST > 0).

Table 129. FMC_BCRx bitfields (mode 1)

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don’t care
5:4	MWID	As needed

Table 129. FMC_BCRx bitfields (mode 1) (continued)

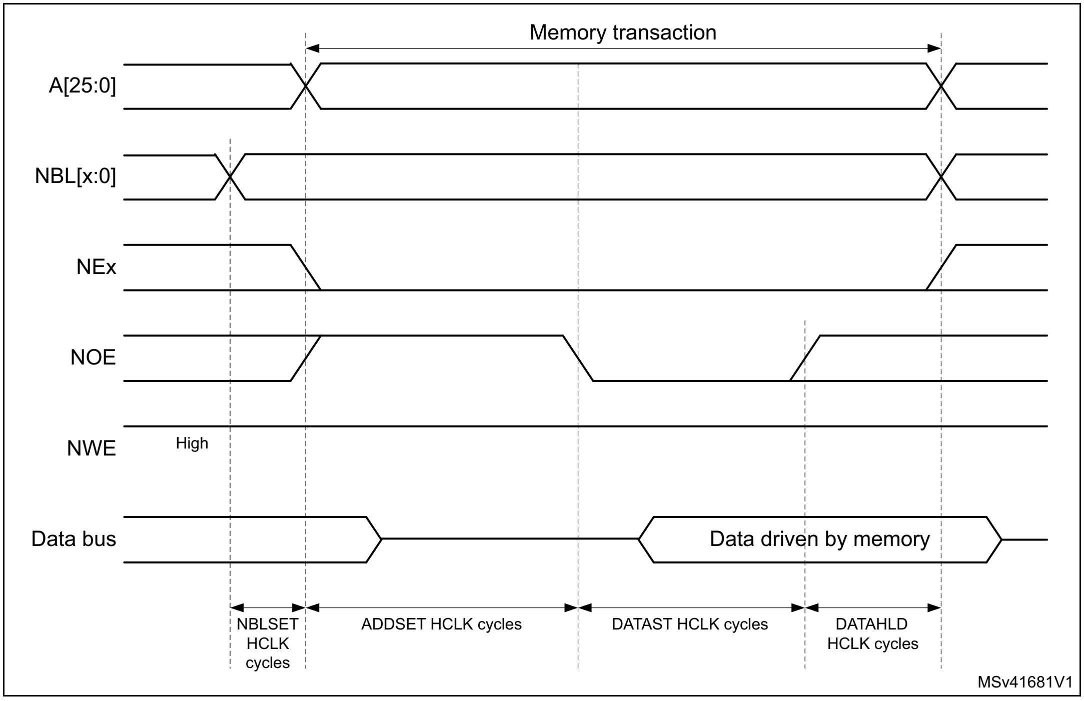
Bit number	Bit name	Value to set
3:2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 130. FMC_BTRx bitfields (mode 1)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses, DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	Don’t care
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles).
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 0.

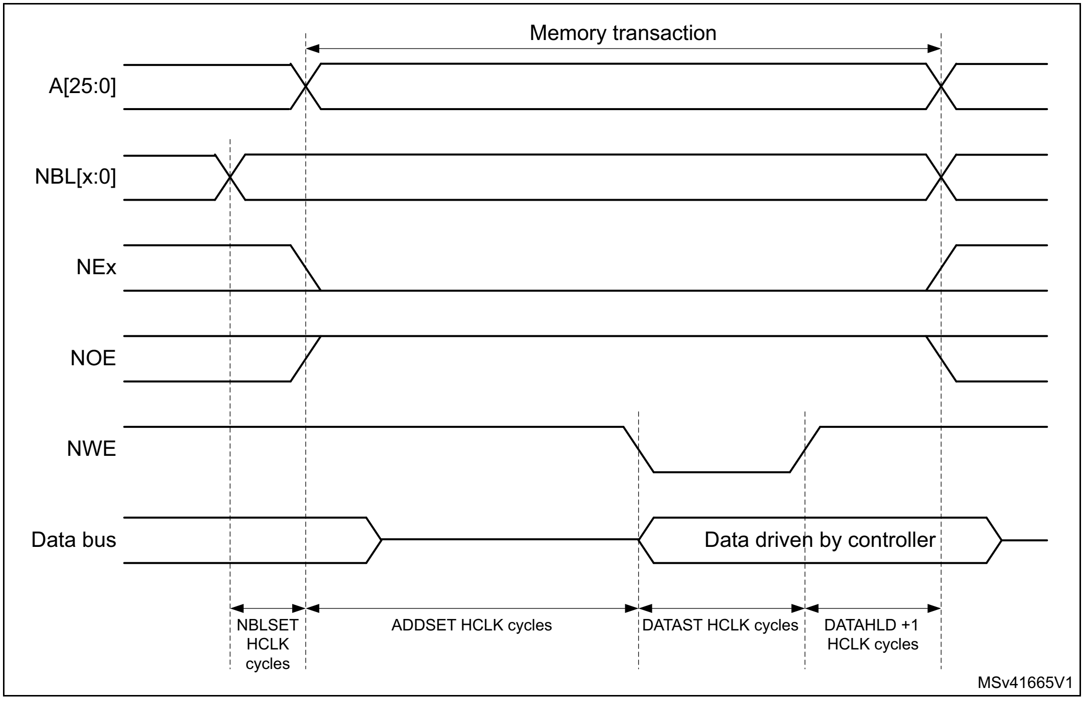
Mode A - SRAM/FRAM/PSRAM (CRAM) OE toggling

Figure 56. Mode A read access waveforms



1. NBL[1:0] are driven low during the read access

Figure 57. Mode A write access waveforms



The differences compared with Mode 1 are the toggling of NOE and the independent read and write timings.

Table 131. FMC_BCRx bitfields (mode A)

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don’t care
5:4	MWID	As needed

Table 131. FMC_BCRx bitfields (mode A) (continued)

Bit number	Bit name	Value to set
3:2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 132. FMC_BTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).
29:28	ACCMOD	0x0
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 133. FMC_BWTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x0
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for write accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Mode 2/B - NOR Flash

Figure 58. Mode 2 and mode B read access waveforms

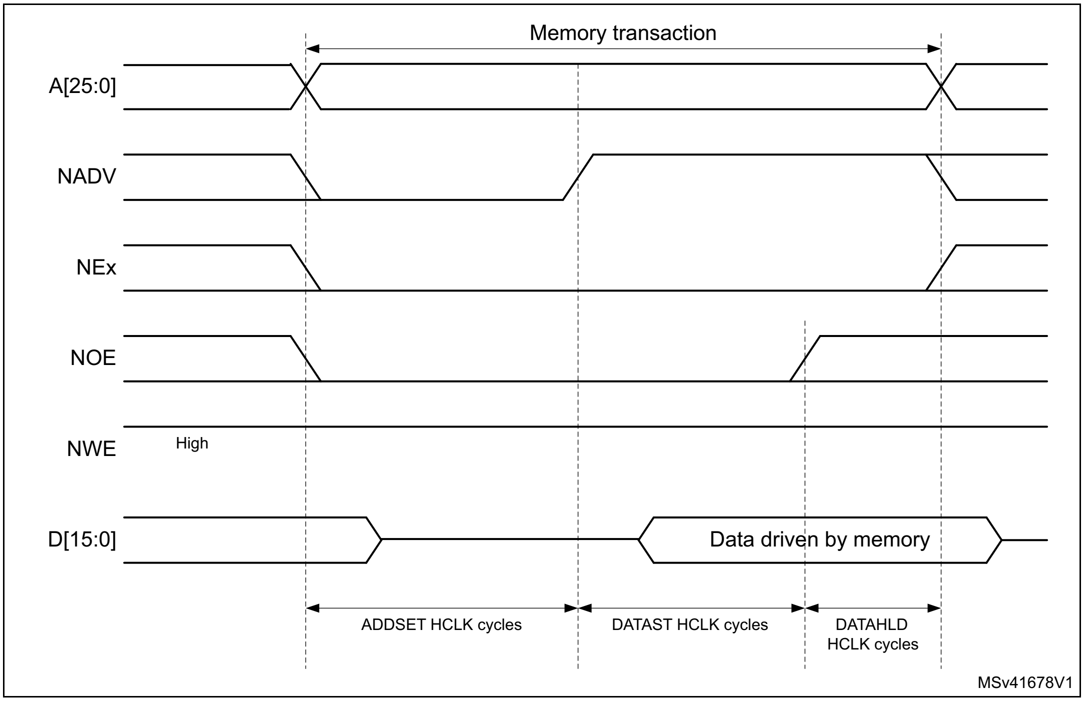


Figure 59. Mode 2 write access waveforms

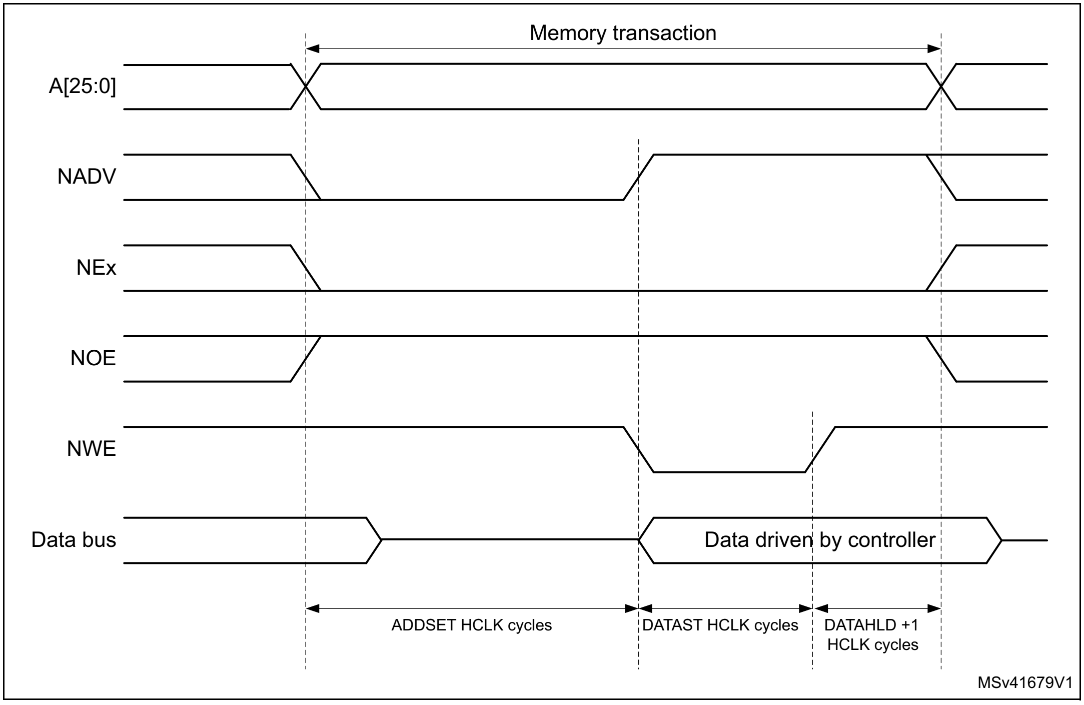
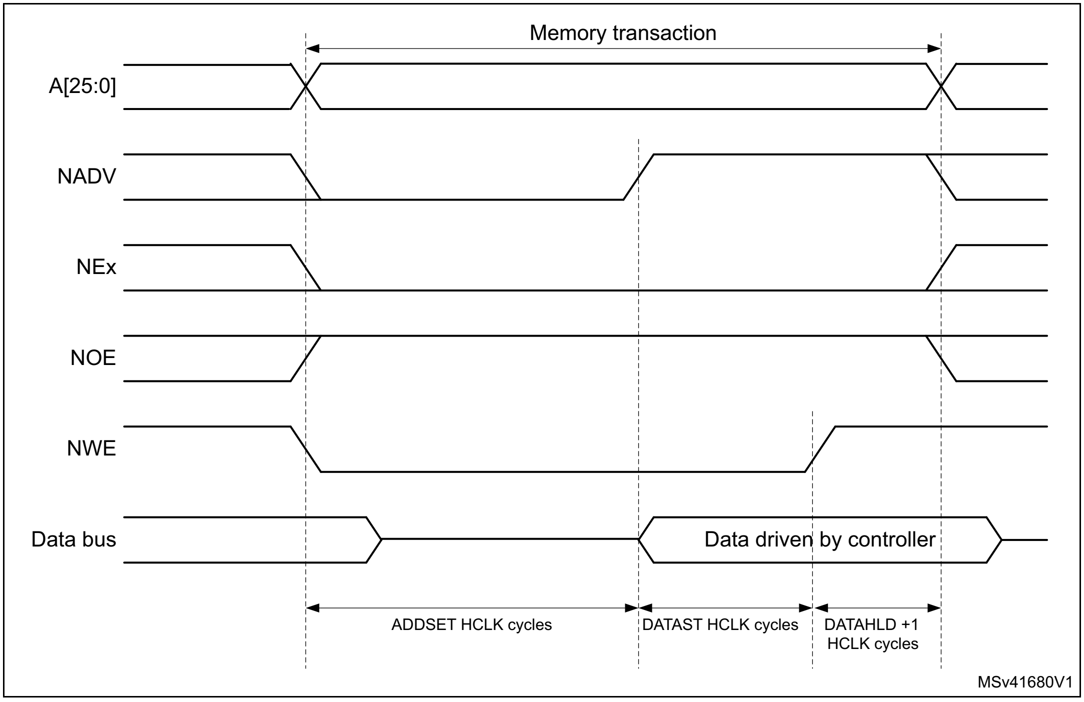


Figure 60. Mode B write access waveforms



The differences with mode 1 are the toggling of NWE and the independent read and write timings when extended mode is set (mode B).

Table 134. FMC_BCRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	Don't care
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed

Table 134. FMC_BCRx bitfields (mode 2/B) (continued)

Bit number	Bit name	Value to set
3:2	MTYP	0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 135. FMC_BTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses and DATAHLD+1 HCLK cycles for write accesses when Extended mode is disabled).
29:28	ACCMOD	0x1 if Extended mode is set
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the access second phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 136. FMC_BWTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x1 if Extended mode is set
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the access second phase (DATAST HCLK cycles) for write accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Note: The FMC_BWTRx register is valid only if the Extended mode is set (mode B), otherwise its content is don’t care.

Mode C - NOR Flash - OE toggling

Figure 61. Mode C read access waveforms

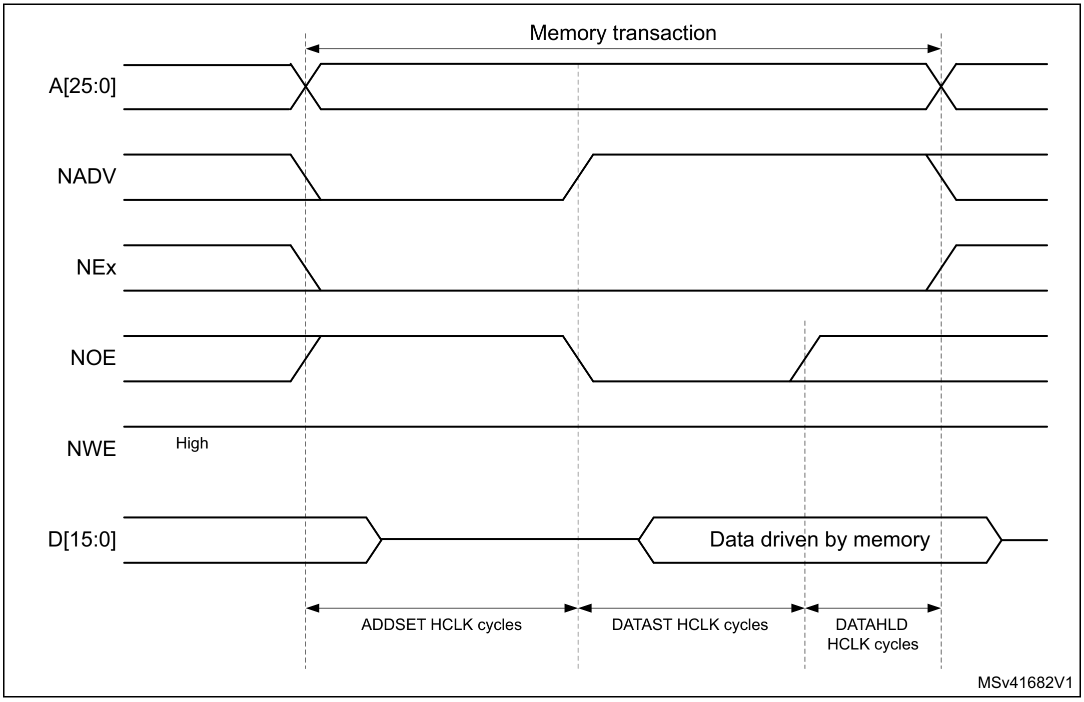
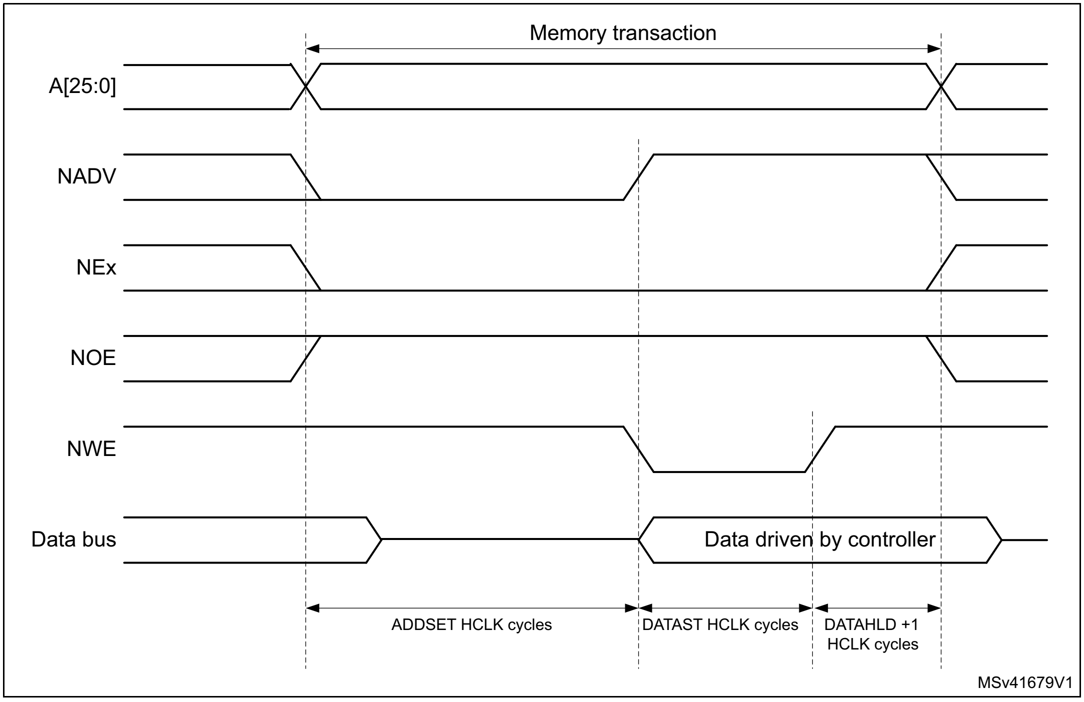


Figure 62. Mode C write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 137. FMC_BCRx bitfields (mode C)

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	Don’t care
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x02 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 138. FMC_BTRx bitfields (mode C)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).
29:28	ACCMOD	0x2
27:24	DATLAT	0x0
23:20	CLKDIV	0x0
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 139. FMC_BWTRx bitfields (mode C)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x2
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for write accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 63. Mode D read access waveforms

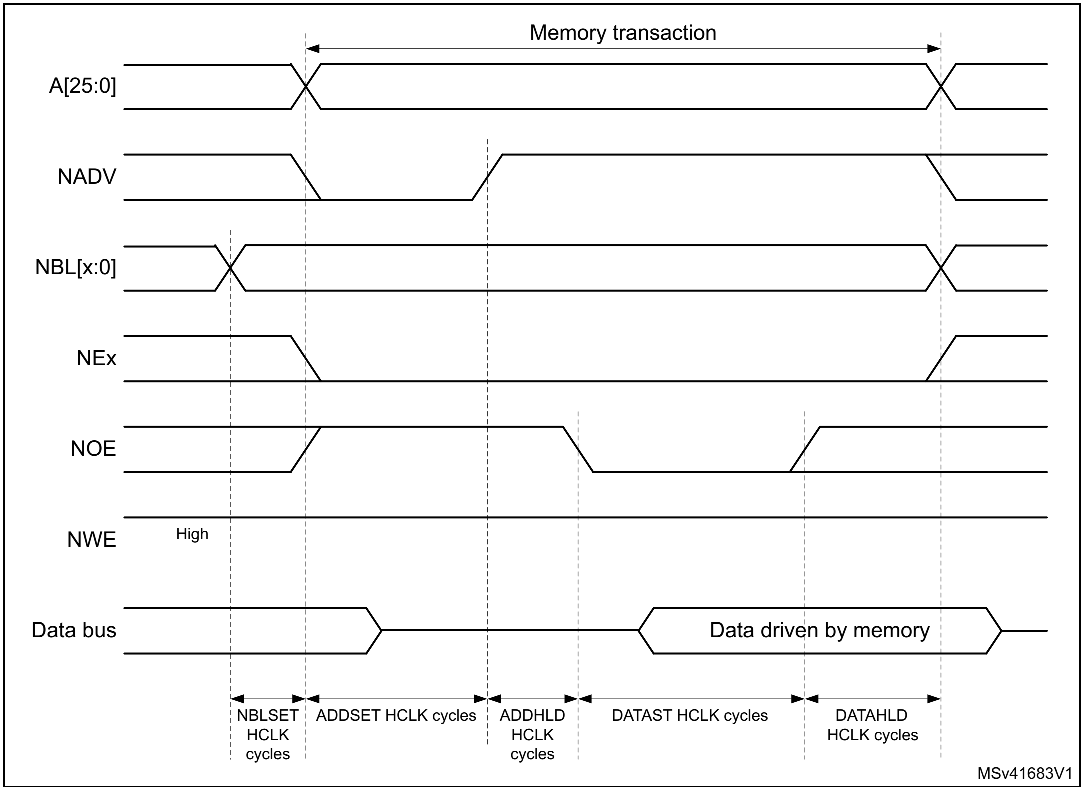
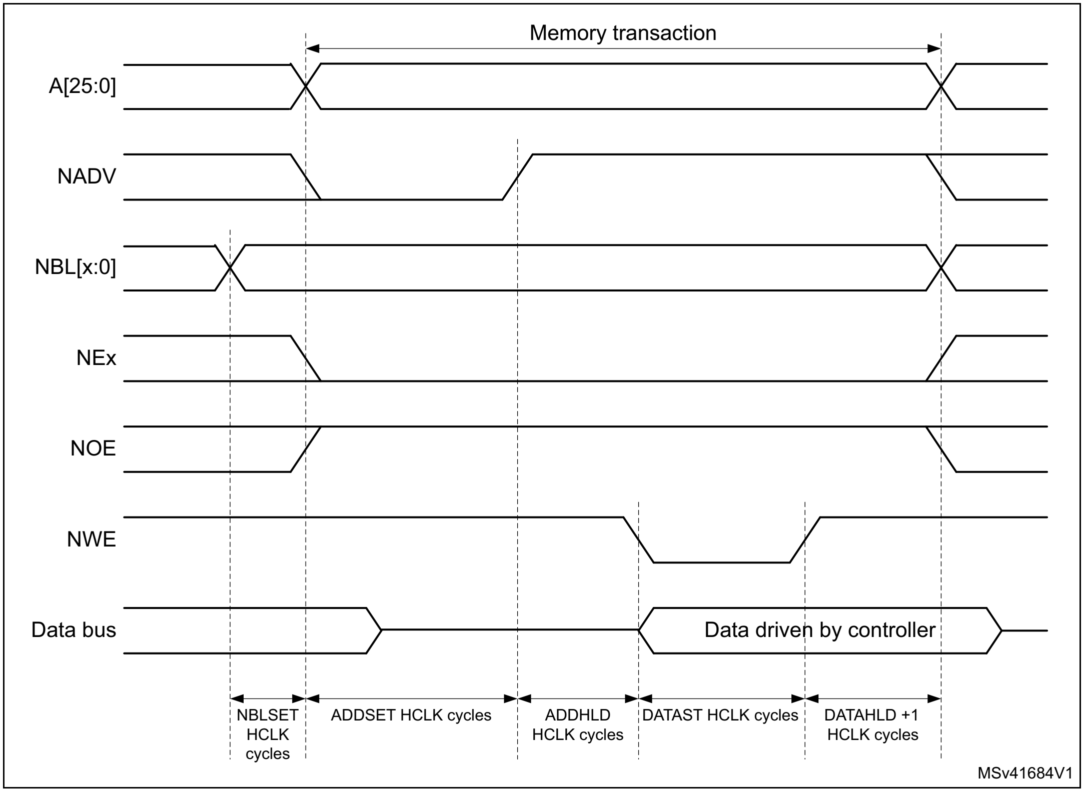


Figure 64. Mode D write access waveforms



The differences with mode 1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 140. FMC_BCRx bitfields (mode D)

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1

Table 140. FMC_BCRx bitfields (mode D) (continued)

Bit number	Bit name	Value to set
6	FACCEN	Set according to memory support
5:4	MWID	As needed
3:2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 141. FMC_BTRx bitfields (mode D)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).
29:28	ACCMOD	0x3
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Duration of the middle phase of the read access (ADDHLD HCLK cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 1.

Table 142. FMC_BWTRx bitfields (mode D)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x3
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles).
7:4	ADDHLD	Duration of the middle phase of the write access (ADDHLD HCLK cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR Flash memory

Figure 65. Muxed read access waveforms

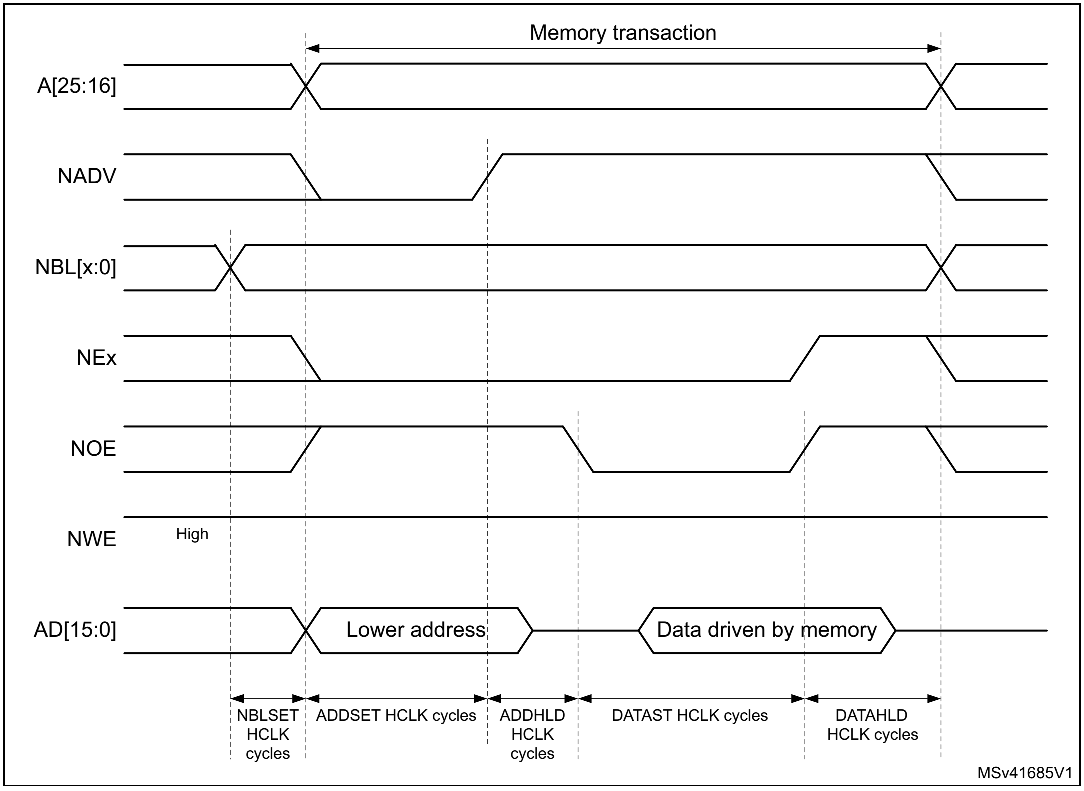
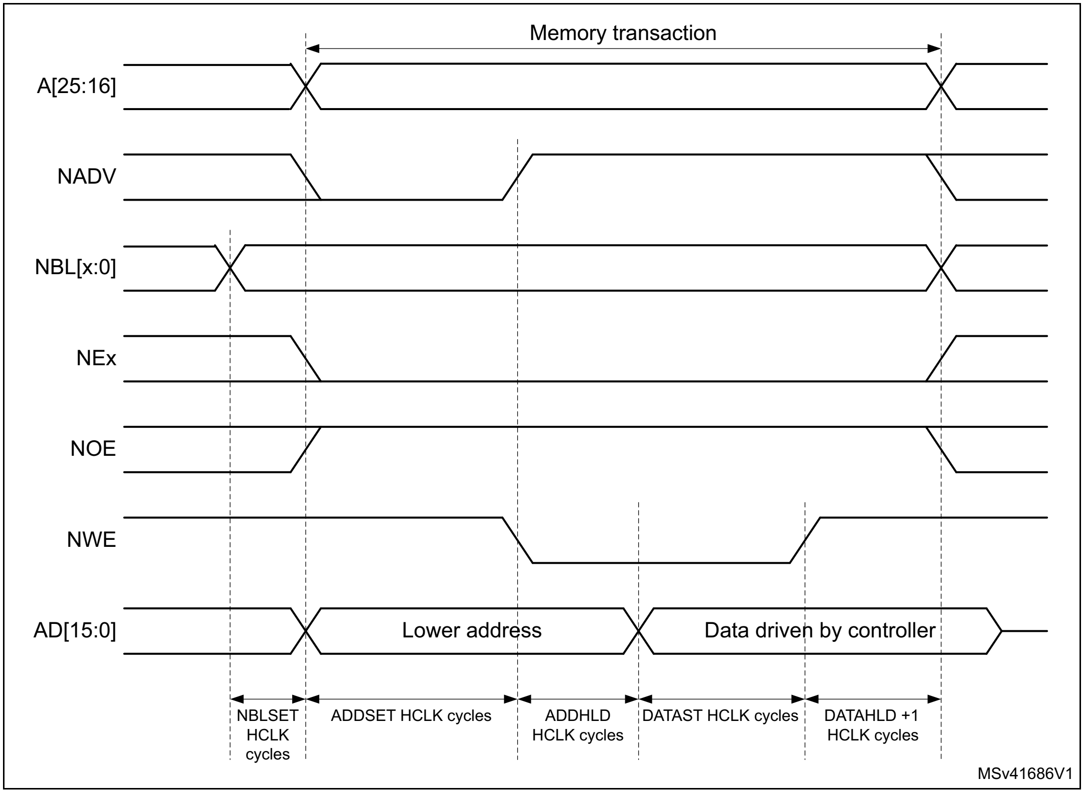


Figure 66. Muxed write access waveforms



The difference with mode D is the drive of the lower address byte(s) on the data bus.

Table 143. FMC_BCRx bitfields (Muxed mode)

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1

Table 143. FMC_BCRx bitfields (Muxed mode) (continued)

Bit number	Bit name	Value to set
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR Flash memory) or 0x1(PSRAM)
1	MUXEN	0x1
0	MBKEN	0x1

Table 144. FMC_BTRx bitfields (Muxed mode)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses, DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x0
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles).
7:4	ADDHLD	Duration of the middle phase of the access (ADDHLD HCLK cycles).
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 1.

WAIT management in asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected 4 HCLK cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:

$$\text{DATAST} \geq (4 \times \text{HCLK}) + \text{max\_wait\_assertion\_time}$$

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   if

$$\text{max\_wait\_assertion\_time} > \text{address\_phase} + \text{hold\_phase}$$

   then:

$$\text{DATAST} \geq (4 \times \text{HCLK}) + (\text{max\_wait\_assertion\_time} - \text{address\_phase} - \text{hold\_phase})$$

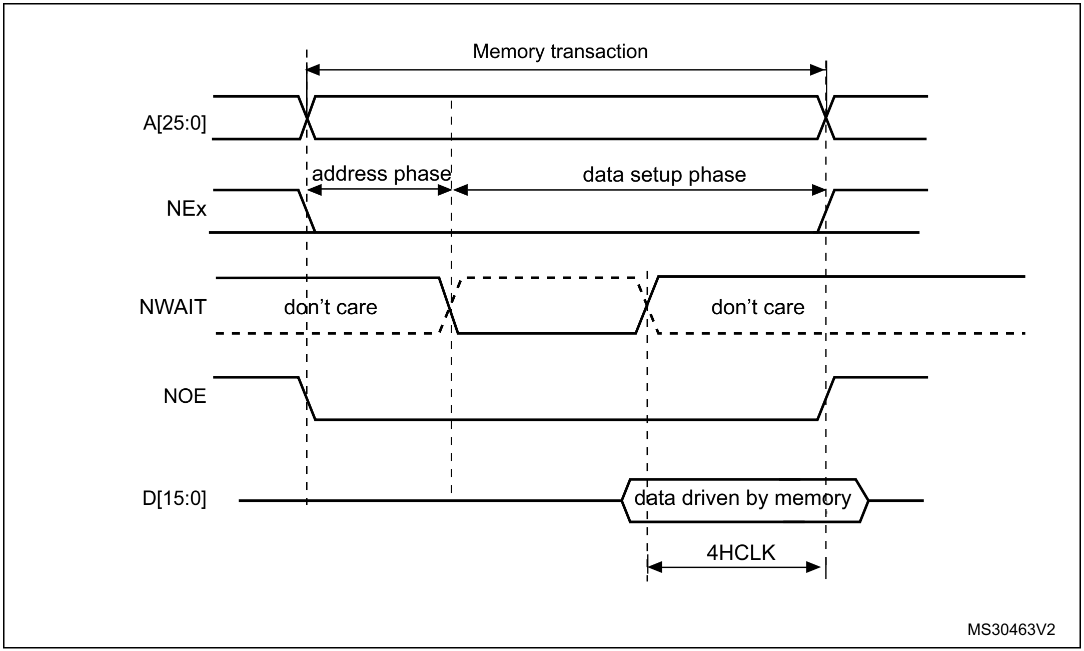
   otherwise

$$\text{DATAST} \geq 4 \times \text{HCLK}$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

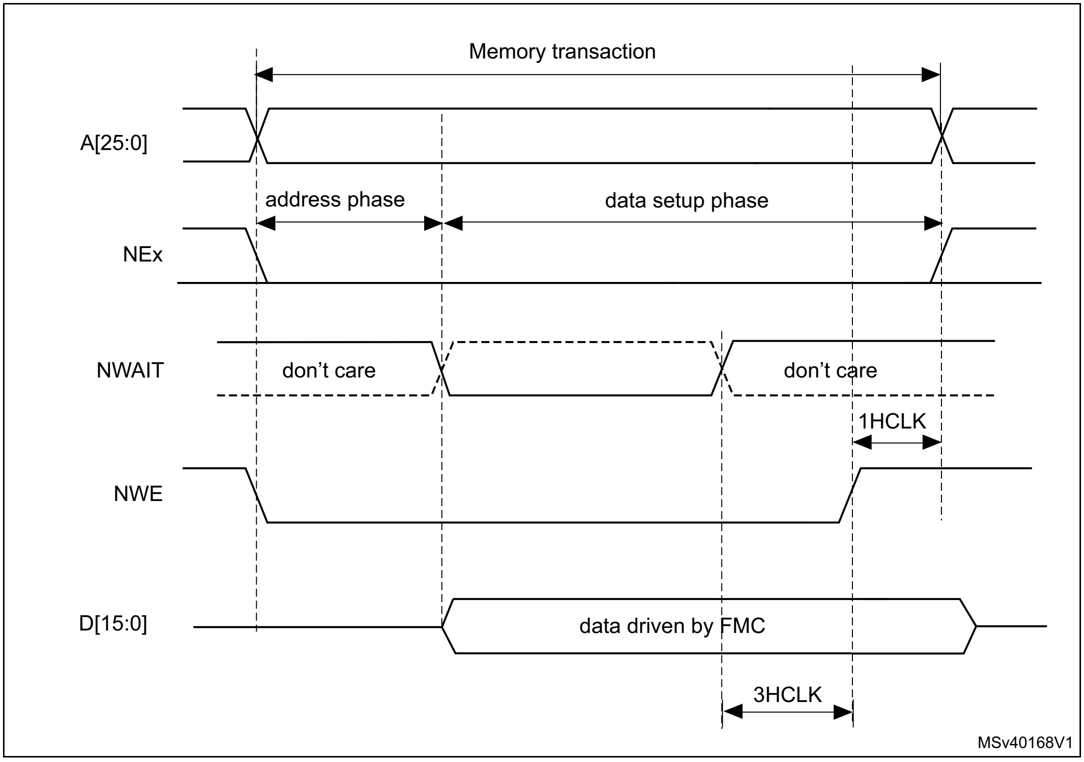
Figure 67 and *Figure 68* show the number of HCLK clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 67. Asynchronous wait during a read access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 68. Asynchronous wait during a write access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

CellularRAM™ (PSRAM) refresh management

The CellularRAM™ does not allow maintaining the chip select signal (NE) low for longer than the t_{CEM} timing specified for the memory device. This timing can be programmed in the FMC_PCSCNTR register. It defines the maximum duration of the NE low pulse in HCLK cycles for asynchronous accesses and FMC_CLK cycles for synchronous accesses

19.6.5 Synchronous transactions

The memory clock, FMC_CLK, is a submultiple of HCLK. It depends on the value of CLKDIV and the MWID/ AHB data size, following the formula given below:

Whatever MWID size: 16 or 8-bit, the FMC_CLK divider ratio is always defined by the programmed CLKDIV value.

Example:

- If CLKDIV=1, MWID = 16 bits, AHB data size=8 bits, FMC_CLK=HCLK/2.

NOR Flash memories specify a minimum time from NADV assertion to CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

Data latency versus NOR memory latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR Flash configuration register. The FMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR Flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR Flash latency and the FMC DATLAT parameter can be either:

- NOR Flash latency = (DATLAT + 2) CLK clock cycles
- or NOR Flash latency = (DATLAT + 3) CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected bank is configured in Burst mode for synchronous accesses, if for example an AHB single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AHB transfer is 16 bits), or length 2 (if the AHB transfer is 32 bits) and de-assert the chip select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Cross boundary page for CellularRAM™ 1.5

CellularRAM™ 1.5 does not allow burst access to cross the page boundary. The FMC controller allows to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

Wait management

For synchronous NOR Flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the chip select and output enable signals valid. It does not consider the data as valid.

In Burst mode, there are two timing configurations for the NOR Flash NWAIT signal:

- The Flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset).
- The Flash memory asserts the NWAIT signal during the wait state

The FMC supports both NOR Flash wait state configurations, for each chip select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0..3).

Figure 69. Wait configuration waveforms

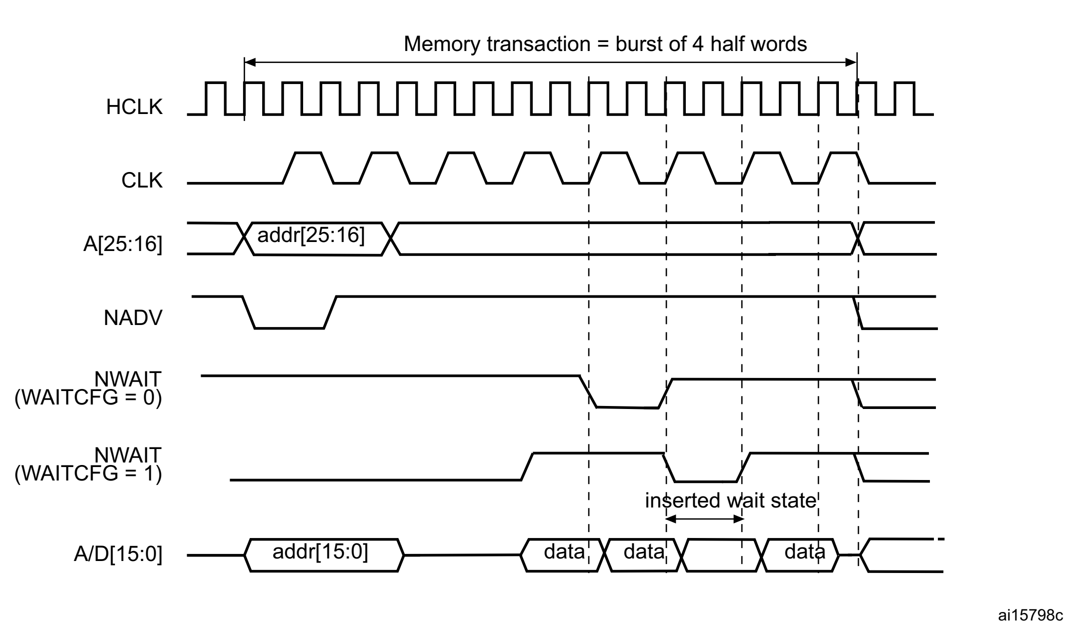
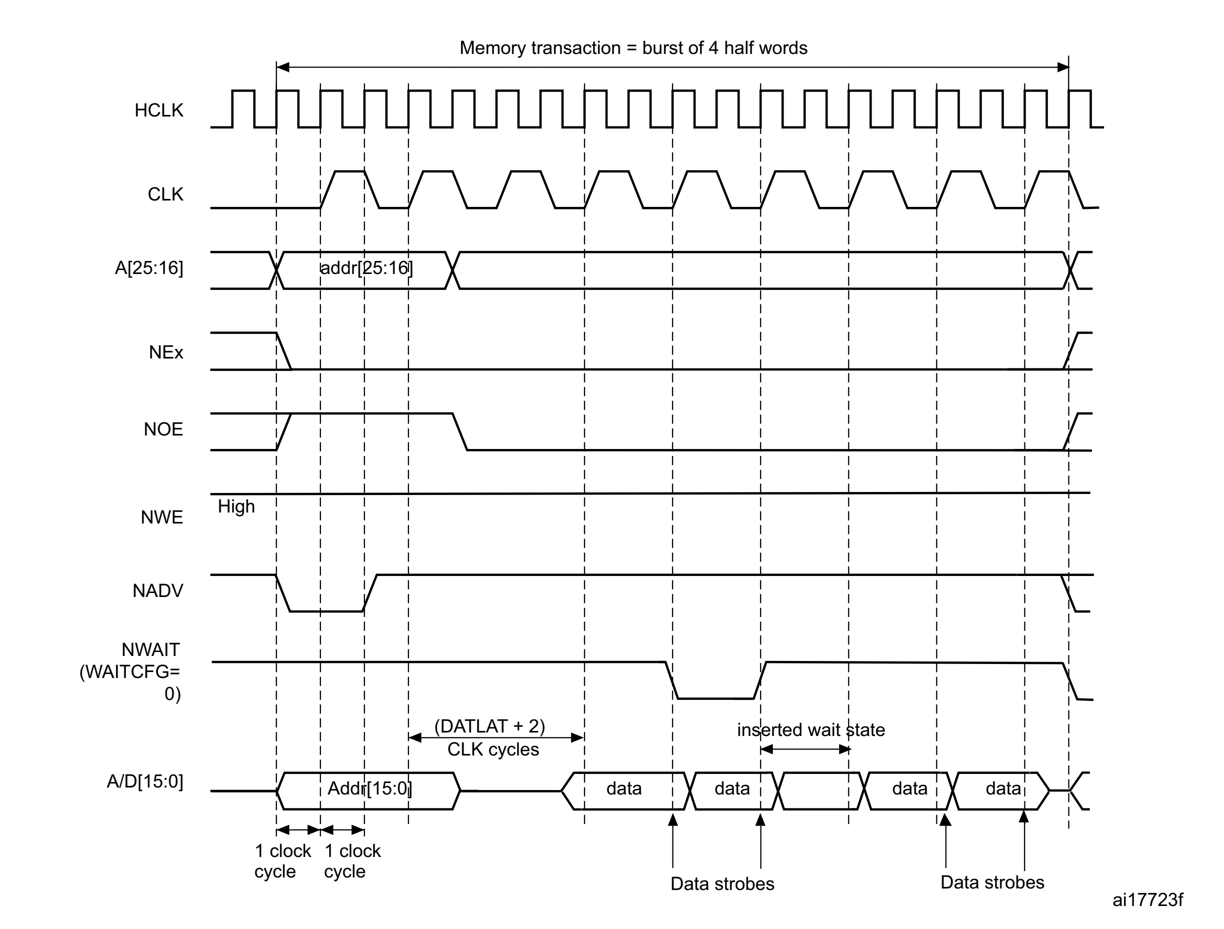


Figure 70. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)



1. Byte lane outputs (NBL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.

Table 145. FMC_BCRx bitfields (Synchronous multiplexed read mode)

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	Don’t care
20	CCLKEN	As needed
19	CBURSTRW	No effect on synchronous read
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	To be set to 1 if the memory supports this feature, to be kept at 0 otherwise
12	WREN	No effect on synchronous read
11	WAITCFG	To be set according to memory
10	Reserved	0x0

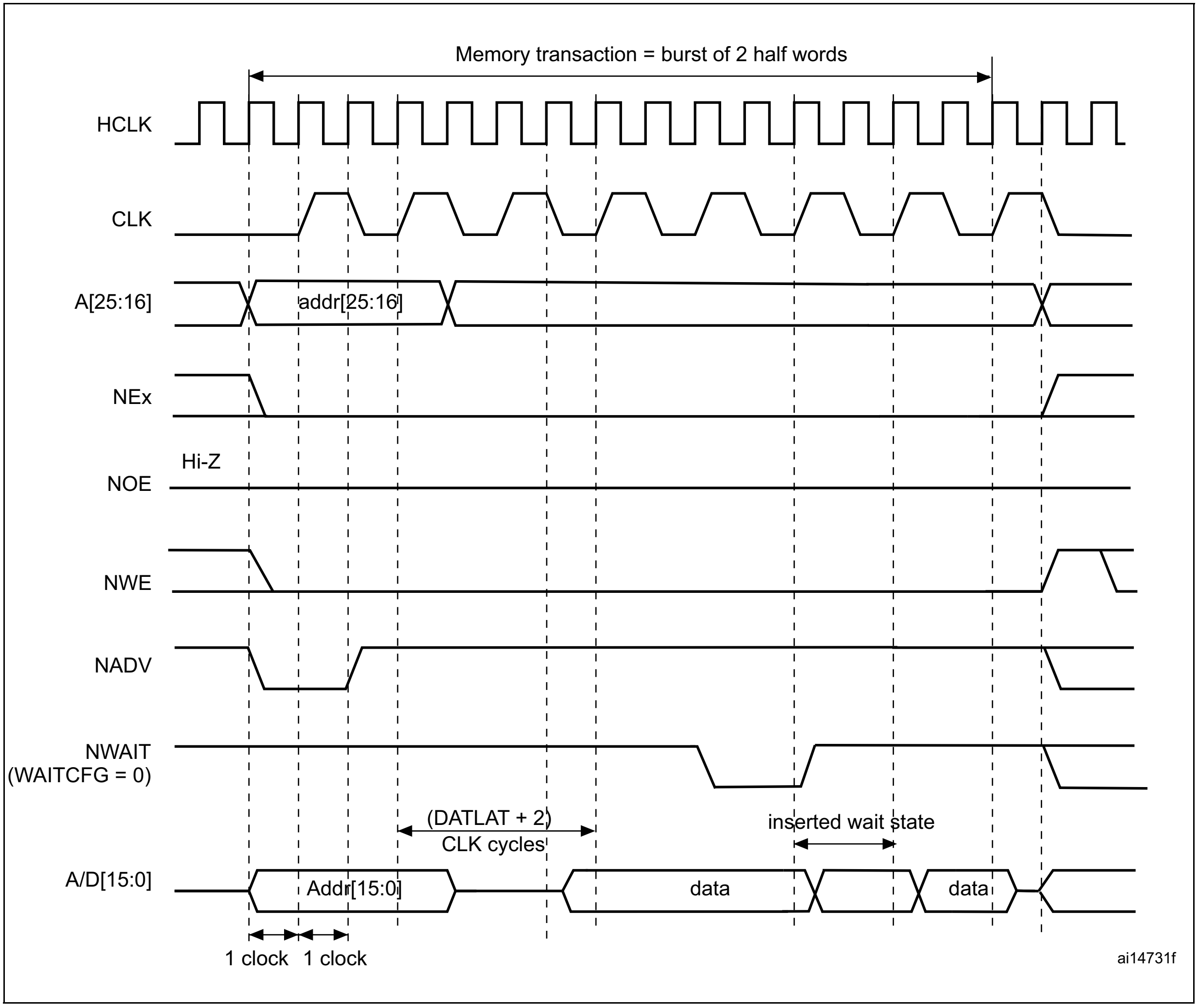
Table 145. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

Bit number	Bit name	Value to set
9	WAITPOL	To be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1
6	FACCEN	Set according to memory support (NOR Flash memory)
5-4	MWID	As needed
3-2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 146. FMC_BTRx bitfields (Synchronous multiplexed read mode)

Bit number	Bit name	Value to set
31:30	DATAHLD	Don’t care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK 0x1 to get CLK = 2 × HCLK ..
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15-8	DATAST	Don’t care
7-4	ADDHLD	Don’t care
3-0	ADDSET	Don’t care

Figure 71. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)



1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 147. FMC_BCRx bitfields (Synchronous multiplexed write mode)

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	Don’t care
20	CCLKEN	As needed
19	CBURSTRW	0x1
18:16	CPSIZE	As needed (0x1 for CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	To be set to 1 if the memory supports this feature, to be kept at 0 otherwise.

Table 147. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

Bit number	Bit name	Value to set
12	WREN	0x1
11	WAITCFG	0x0
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	Reserved	0x1
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 148. FMC_BTRx bitfields (Synchronous multiplexed write mode)

Bit number	Bit name	Value to set
31-30	DATAHLD	Don’t care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK 0x1 to get CLK = 2 × HCLK
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15-8	DATAST	Don’t care
7-4	ADDHLD	Don’t care
3-0	ADDSET	Don’t care

19.6.6 NOR/PSRAM controller registers

SRAM/NOR-Flash chip-select control register for bank x (FMC_BCRx) (x = 1 to 4)

Address offset: 8 * (x – 1), (x = 1 to 4)

Reset value: Bank 1: 0x0000 30DB

Reset value: Bank 2: 0x0000 30D2

Reset value: Bank 3: 0x0000 30D2

Reset value: Bank 4: 0x0000 30D2

This register contains the control information of each memory bank, used for SRAMs, PSRAM, FRAM and NOR Flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		WFDIS	CCLK EN	CBURST RW	CPSIZE[2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNC WAIT	EXT MOD	WAIT EN	WREN	WAIT CFG	Res.	WAIT POL	BURST EN	Res.	FACC EN	MWID[1:0]		MTYP[1:0]		MUX EN	MBK EN
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 23:22 **NBLSET[1:0]:** Byte lane (NBL) setup

These bits configure the NBL setup timing from NBLx low to chip select NEx low.

00: NBL setup time is 0 AHB clock cycle

01: NBL setup time is 1 AHB clock cycle

10: NBL setup time is 2 AHB clock cycles

11: NBL setup time is 3 AHB clock cycles

Bit 21 **WFDIS:** Write FIFO disable

This bit disables the Write FIFO used by the FMC controller.

0: Write FIFO enabled (Default after reset)

1: Write FIFO disabled

Note: The WFDIS bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register.

Bit 20 **CCLKEN:** Continuous clock enable

This bit enables the FMC_CLK clock output to external memory devices.

0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is specified by the programmed CLKDIV value in the FMC_BCRx register (default after reset).

1: The FMC_CLK is generated continuously during asynchronous and synchronous access. The FMC_CLK clock is activated when the CCLKEN is set.

Note: The CCLKEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register. Bank 1 must be configured in Synchronous mode to generate the FMC_CLK continuous clock.

Note: If CCLKEN bit is set, the FMC_CLK clock ratio is specified by CLKDIV value in the FMC_BTR1 register. CLKDIV in FMC_BWTR1 is don't care.

Note: If the Synchronous mode is used and CCLKEN bit is set, the synchronous memories connected to other banks than Bank 1 are clocked by the same clock (the CLKDIV value in the FMC_BTR2..4 and FMC_BWTR2..4 registers for other banks has no effect.)

Bit 19 **CBURSTRW:** Write burst enable

For PSRAM (CRAM) operating in Burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.

0: Write operations are always performed in Asynchronous mode.

1: Write operations are performed in Synchronous mode.

Bits 18:16 **CPSIZE[2:0]**: CRAM page size

These are used for CellularRAM™ 1.5 which does not allow burst access to cross the address boundaries between pages. When these bits are configured, the FMC controller splits automatically the burst access when the memory page size is reached (refer to memory datasheet for page size).

000: No burst split when crossing page boundary (default after reset)

001: 128 bytes

010: 256 bytes

011: 512 bytes

100: 1024 bytes

Others: reserved

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables/disables the FMC to use the wait signal even during an asynchronous protocol.

0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset).

1: NWAIT signal is taken in to account when running an asynchronous protocol.

Bit 14 **EXTMOD:** Extended mode enable

This bit enables the FMC to program the write timings for non multiplexed asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.

0: values inside FMC_BWTR register are not taken into account (default after reset)

1: values inside FMC_BWTR register are taken into account

Note: When the Extended mode is disabled, the FMC can operate in mode 1 or mode 2 as follows:

- *Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01)*
- *Mode 2 is the default mode when the NOR memory type is selected (MTYP = 0x10).*

Bit 13 **WAITEN:** Wait enable bit

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in Synchronous mode.

0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed Flash latency period).

1: NWAIT signal is enabled (its level is taken into account after the programmed latency period to insert wait states if asserted) (default after reset).

Bit 12 **WREN:** Write enable bit

This bit indicates whether write operations are enabled/disabled in the bank by the FMC.

0: Write operations are disabled in the bank by the FMC, an AHB error is reported.

1: Write operations are enabled for the bank by the FMC (default after reset).

Bit 11 **WAITCFG:** Wait timing configuration

The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the memory in Synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:

0: NWAIT signal is active one data cycle before wait state (default after reset).

1: NWAIT signal is active during wait state (not used for PSRAM).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **WAITPOL:** Wait signal polarity bit

Defines the polarity of the wait signal from memory used for either in Synchronous or Asynchronous mode.

0: NWAIT active low (default after reset)

1: NWAIT active high

Bit 8 **BURSTEN:** Burst enable bit

This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in Burst mode.

0: Burst mode disabled (default after reset). Read accesses are performed in Asynchronous mode.

1: Burst mode enable. Read accesses are performed in Synchronous mode.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FACCEN:** Flash access enable

Enables NOR Flash memory access operations.

0: Corresponding NOR Flash memory access is disabled.

1: Corresponding NOR Flash memory access is enabled (default after reset).

Bits 5:4 **MWID[1:0]:** Memory data bus width

Defines the external memory device width, valid for all type of memories.

00: 8 bits

01: 16 bits (default after reset)

10: reserved

11: reserved

Bits 3:2 **MTYP[1:0]:** Memory type

Defines the type of external memory attached to the corresponding memory bank.
00: SRAM/FRAM (default after reset for Bank 2...4)
01: PSRAM (CRAM) / FRAM
10: NOR Flash/OneNAND Flash (default after reset for Bank 1)
11: reserved

Bit 1 **MUXEN:** Address/data multiplexing enable bit

When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:
0: Address/data non multiplexed
1: Address/data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory bank enable bit

Enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AHB bus.
0: Corresponding memory bank is disabled.
1: Corresponding memory bank is enabled.

SRAM/NOR-Flash chip-select timing register for bank x (FMC_BTRx)

Address offset: 0x04 + 8 * (x – 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR Flash memories.If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[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
DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DATAHLD[1:0]:** Data hold phase duration

These bits are written by software to define the duration of the data hold phase in HCLK cycles (refer to *Figure 54* to *Figure 66*), used in asynchronous accesses:
For read accesses
00: DATAHLD phase duration = 0 × HCLK clock cycle (default)
01: DATAHLD phase duration = 1 × HCLK clock cycle
10: DATAHLD phase duration = 2 × HCLK clock cycle
11: DATAHLD phase duration = 3 × HCLK clock cycle
For write accesses
00: DATAHLD phase duration = 1 × HCLK clock cycle (default)
01: DATAHLD phase duration = 2 × HCLK clock cycle
10: DATAHLD phase duration = 3 × HCLK clock cycle
11: DATAHLD phase duration = 4 × HCLK clock cycle

Bits 29:28 **ACCMOD[1:0]:** Access mode

Specifies the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: Access mode A

01: Access mode B

10: Access mode C

11: Access mode D

Bits 27:24 **DATLAT[3:0]:** (see note below bit descriptions): Data latency for synchronous memory

For synchronous access with read/write Burst mode enabled (BURSTEN / CBURSTRW bits set), defines the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data:

This timing parameter is not expressed in HCLK periods, but in FMC_CLK periods.

For asynchronous access, this value is don't care.

0000: Data latency of 2 CLK clock cycles for first burst access

1111: Data latency of 17 CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)

Defines the period of FMC_CLK clock output signal, expressed in number of HCLK cycles:

0000: FMC_CLK period= 1x HCLK period

0001: FMC_CLK period = 2 × HCLK periods

0010: FMC_CLK period = 3 × HCLK periods

1111: FMC_CLK period = 16 × HCLK periods (default value after reset)

In asynchronous NOR Flash, SRAM or PSRAM accesses, this value is don’t care.

Note: Refer to Section 19.6.5: Synchronous transactions for FMC_CLK divider ratio formula)

Bits 19:16 **BUSTURN[3:0]:** Bus turnaround phase duration

These bits are written by software to add a delay at the end of current read or write transaction to next transaction on the same bank.

This delay allows to match the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access (t_{EHQZ}, chip enable high to output Hi-Z). This delay is recommended for mode D and muxed mode. For non-muxed memory, the bus turnaround delay can be set to minimum value.

(BUSTURN + 1)HCLK period ≥ max(t_{EHEL} min, t_{EHQZ} max)

For FRAM memories, the bus turnaround delay should be configured to match the minimum tPC (precharge time) timings. The bus turnaround delay is inserted between any consecutive transactions on the same bank (read/read, write/write, read/write and write/read) to match the tPC memory timing. The chip select is toggling between any consecutive accesses.

(BUSTURN + 1)HCLK period ≥ t_{PC} min

0000: BUSTURN phase duration = 1 HCLK clock cycle added

...

1111: BUSTURN phase duration = 16 x HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration

These bits are written by software to define the duration of the data phase (refer to *Figure 54* to *Figure 66*), used in asynchronous accesses:

0000 0000: Reserved
0000 0001: DATAST phase duration = 1 × HCLK clock cycles
0000 0010: DATAST phase duration = 2 × HCLK clock cycles
...
1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

For each memory type and access mode data-phase duration, refer to the respective figure (*Figure 54* to *Figure 66*).

Example: Mode 1, write access, DATAST=1: Data-phase duration= DATAST+1 = 2 HCLK clock cycles.

Note: In synchronous accesses, this value is don't care.

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 54* to *Figure 66*), used in mode D or multiplexed accesses:

0000: Reserved
0001: ADDHLD phase duration =1 × HCLK clock cycle
0010: ADDHLD phase duration = 2 × HCLK clock cycle
...
1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

For each access mode address-hold phase duration, refer to the respective figure (*Figure 54* to *Figure 66*).

Note: In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 54* to *Figure 66*), used in SRAMs, ROMs, asynchronous NOR Flash and PSRAM:

0000: ADDSET phase duration = 0 × HCLK clock cycle
...
1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)

For each access mode address setup phase duration, refer to the respective figure (*Figure 54* to *Figure 66*).

Note: In synchronous accesses, this value is don't care.

In Muxed mode or mode D, the minimum value for ADDSET is 1.

In mode 1 and PSRAM memory, the minimum value for ADDSET is 1.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to prolong the latency as needed.
With PSRAMs (CRAMs) the filled DATLAT must be set to 0, so that the FMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.
This method can be used also with the latest generation of synchronous Flash memories that issue the NWAIT signal, unlike older Flash memories (check the datasheet of the specific Flash memory being used).

SRAM/NOR-Flash write timing registers x (FMC_BWTRx)

Address offset: 0x104 + 8 * (x – 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank. It is used for SRAMs, PSRAMs and NOR Flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[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
DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DATAHLD[1:0]:** Data hold phase duration

These bits are written by software to define the duration of the data hold phase in HCLK cycles (refer to *Figure 54* to *Figure 66*), used in asynchronous write accesses:

00: DATAHLD phase duration = 1 × HCLK clock cycle (default)
01: DATAHLD phase duration = 2 × HCLK clock cycle
10: DATAHLD phase duration = 3 × HCLK clock cycle
11: DATAHLD phase duration = 4 × HCLK clock cycle

Bits 29:28 **ACCMOD[1:0]:** Access mode.

Specifies the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: Access mode A
01: Access mode B
10: Access mode C
11: Access mode D

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of current write transaction to next transaction on the same bank.

For FRAM memories, the bus turnaround delay should be configured to match the minimum t_{PC} (precharge time) timings. The bus turnaround delay is inserted between any consecutive transactions on the same bank (read/read, write/write, read/write and write/read). The chip select is toggling between any consecutive accesses.

(BUSTURN + 1)HCLK period ≥ tPC min

0000: BUSTURN phase duration = 1 HCLK clock cycle added
...
1111: BUSTURN phase duration = 16 x HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to *Figure 54* to *Figure 66*), used in asynchronous SRAM, PSRAM and NOR Flash memory accesses:

0000 0000: Reserved
0000 0001: DATAST phase duration = 1 × HCLK clock cycles
0000 0010: DATAST phase duration = 2 × HCLK clock cycles
...
1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration.

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 63* to *Figure 66*), used in asynchronous multiplexed accesses:

0000: Reserved
0001: ADDHLD phase duration = 1 × HCLK clock cycle
0010: ADDHLD phase duration = 2 × HCLK clock cycle
...
1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous NOR Flash accesses, this value is not used, the address hold phase is always 1 Flash clock period duration.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration.

These bits are written by software to define the duration of the *address setup* phase in HCLK cycles (refer to *Figure 54* to *Figure 66*), used in asynchronous accesses:

0000: ADDSET phase duration = 0 × HCLK clock cycle
...
1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous accesses, this value is not used, the address setup phase is always 1 Flash clock period duration. In muxed mode, the minimum ADDSET value is 1.

PSRAM chip select counter register (FMC_PCSCNTR)

Address offset: 0x20

Reset value: 0x0000 0000

This register contains the PSRAM chip select counter value for Synchronous and Asynchronous modes. The chip select counter is common to all banks and can be enabled separately on each bank. During PSRAM read or write accesses, this value is loaded into a timer which is decremented while the NE signal is held low. When the timer reaches 0, the PSRAM controller splits the current access, toggles NE to allow PSRAM device refresh, and restarts a new access. The programmed counter value guarantees a maximum NE pulse width (t_{CEM}) as specified for PSRAM devices. The counter is reloaded and starts decrementing each time a new access is started by a transition of NE from high to low.

31	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.	CNTB4EN	CNTB3EN	CNTB2EN	CNTB1EN
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSCOUNT[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.

Bit 19 **CNTB4EN:** Counter Bank 4 enable

This bit enables the chip select counter for PSRAM/NOR Bank 4.
0: Counter disabled for Bank 4
1: Counter enabled for Bank 4

Bit 18 **CNTB3EN:** Counter Bank 3 enable

This bit enables the chip select counter for PSRAM/NOR Bank 3.
0: Counter disabled for Bank 3.
1: Counter enabled for Bank 3

Bit 17 **CNTB2EN:** Counter Bank 2 enable

This bit enables the chip select counter for PSRAM/NOR Bank 2.
0: Counter disabled for Bank 2
1: Counter enabled for Bank 2

Bit 16 **CNTB1EN:** Counter Bank 1 enable

This bit enables the chip select counter for PSRAM/NOR Bank 1.
0: Counter disabled for Bank 1
1: Counter enabled for Bank 1

Bits 15:0 **CSCOUNT[15:0]:** Chip select counter.

These bits are written by software to define the maximum chip select low pulse duration. It is expressed in FMC_CLK cycles for synchronous accesses and in HCLK cycles for asynchronous accesses.
The counter is disabled if the programmed value is 0.

19.7 NAND Flash controller

The FMC generates the appropriate signal timings to drive the following types of device:

- 8- and 16-bit NAND Flash memories

The NAND bank is configured through dedicated registers (*Section 19.7.7*). The programmable memory parameters include access timings (shown in *Table 149*) and ECC configuration.

Table 149. Programmable NAND Flash access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (HCLK) required to set up the address before the command assertion	Read/Write	AHB clock cycle (HCLK)	1	255
Memory wait	Minimum duration (in HCLK clock cycles) of the command assertion	Read/Write	AHB clock cycle (HCLK)	2	255

Table 149. Programmable NAND Flash access parameters (continued)

Parameter	Function	Access mode	Unit	Min.	Max.
Memory hold	Number of clock cycles (HCLK) during which the address must be held (as well as the data if a write access is performed) after the command de-assertion	Read/Write	AHB clock cycle (HCLK)	1	254
Memory databus high-Z	Number of clock cycles (HCLK) during which the data bus is kept in high-Z state after a write access has started	Write	AHB clock cycle (HCLK)	1	255

19.7.1 External memory interface signals

The following tables list the signals that are typically used to interface NAND Flash memory.

Note: The prefix “N” identifies the signals which are active low.

8-bit NAND Flash memory

Table 150. 8-bit NAND Flash

FMC signal name	I/O	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus
NCE	O	Chip select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT	I	NAND Flash ready/busy input signal to the FMC

Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

16-bit NAND Flash memory

Table 151. 16-bit NAND Flash

FMC signal name	I/O	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCE	O	Chip select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT	I	NAND Flash ready/busy input signal to the FMC

Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

19.7.2 NAND Flash supported memories and transactions

Table 152 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND Flash controller are shown in gray.

Table 152. Supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	-
	Asynchronous	W	8	8	Y	-
	Asynchronous	R	16	8	Y	Split into 2 FMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FMC accesses
	Asynchronous	W	32	8	Y	Split into 4 FMC accesses
NAND 16-bit	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	N	-
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses

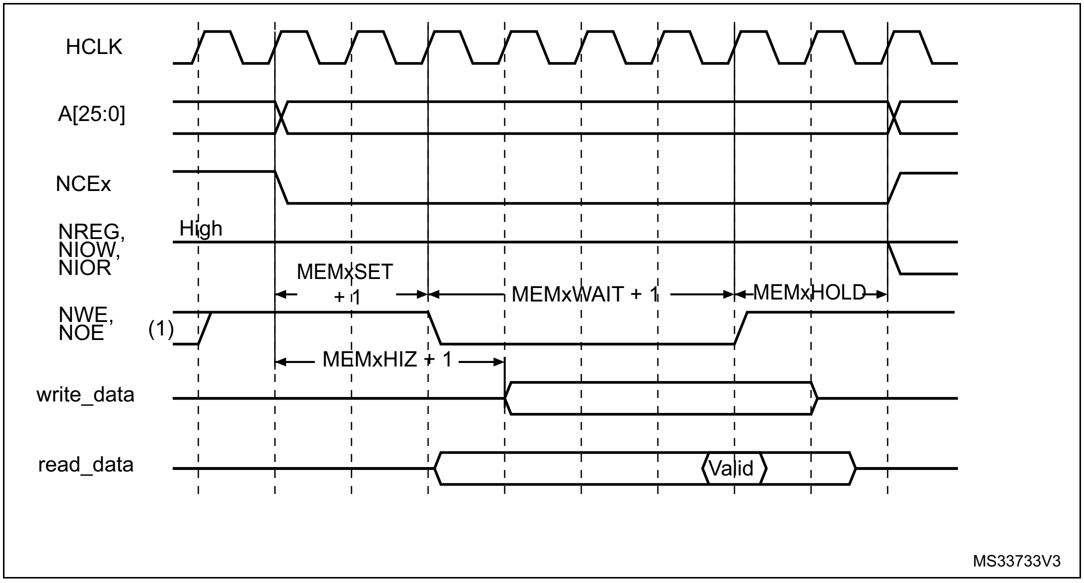
19.7.3 Timing diagrams for NAND Flash memory

The NAND Flash memory bank is managed through a set of registers:

- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_ECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

Each timing configuration register contains three parameters used to define number of HCLK cycles for the three phases of any NAND Flash access, plus one parameter that defines the timing for starting driving the data bus when a write access is performed. *Figure 72* shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

Figure 72. NAND Flash controller waveforms for common memory access



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.
2. For write access, the hold phase delay is (MEMHOLD) HCLK cycles and for read access is (MEMHOLD + 2) HCLK cycles.

19.7.4 NAND Flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND Flash memory device are driven by address signals from the FMC controller. This means that to send a command or an address to the NAND Flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND Flash device requires the following steps:

1. Program and enable the corresponding memory bank by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see *Section 19.7.5: NAND Flash prewait functionality*) registers according to the characteristics of the NAND Flash memory (PWID bits for the data bus width of the NAND Flash, PTYP = 1, PWAITEN = 0 or 1 as needed, see *Section 19.5.2: NAND Flash memory address mapping* for timing configuration).
2. The CPU performs a byte write to the common memory space, with data byte equal to one Flash command byte (for example 0x00 for Samsung NAND Flash devices). The LE input of the NAND Flash memory is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND Flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.
3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] (for 64 Mb x 8 bit NAND Flash memories) in the common memory or attribute space. The ALE input of the NAND Flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it possible to use a different timing configuration of the FMC, which can be used

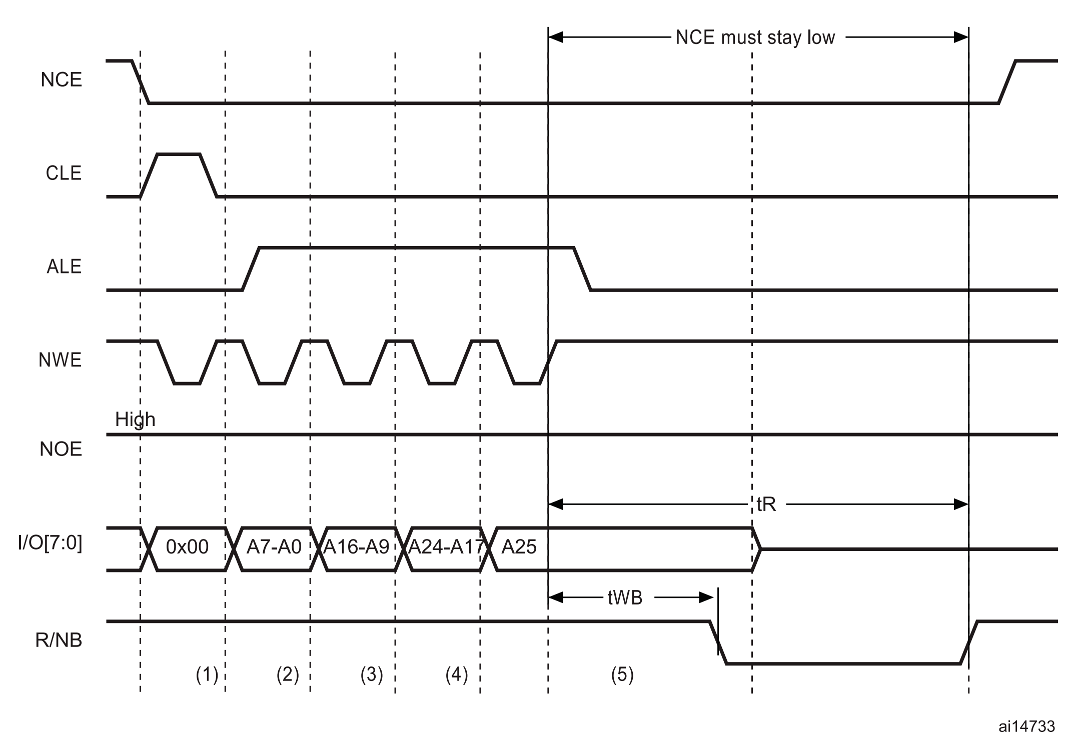
to implement the prewait functionality needed by some NAND Flash memories (see details in *Section 19.7.5: NAND Flash prewait functionality*).

4. The controller waits for the NAND Flash memory to be ready (R/NB signal high), before starting a new access to the same or another memory bank. While waiting, the controller holds the NCE signal active (low).
5. The CPU can then perform byte read operations from the common memory space to read the NAND Flash page (data field + Spare field) byte by byte.
6. The next NAND Flash page can be read without any CPU command or address write operation. This can be done in three different ways:
   - by simply performing the operation described in step 5
   - a new random address can be accessed by restarting the operation at step 3
   - a new command can be sent to the NAND Flash device by restarting at step 2

19.7.5 NAND Flash prewait functionality

Some NAND Flash devices require that, after writing the last part of the address, the controller waits for the R/NB signal to go low. (see *Figure 73*).

Figure 73. Access to non 'CE don't care' NAND-Flash



1. CPU wrote byte 0x00 at address 0x7001 0000.
2. CPU wrote byte A7~A0 at address 0x7002 0000.
3. CPU wrote byte A16~A9 at address 0x7002 0000.
4. CPU wrote byte A24~A17 at address 0x7002 0000.
5. CPU wrote byte A25 at address 0x7802 0000: FMC performs a write access using FMC_PATT timing definition, where ATTHOLD ≥ 7 (providing that (7+1) × HCLK = 112 ns > t_{WB} max). This guarantees that NCE remains low until R/NB goes low and high again (only requested for NAND Flash memories where NCE is not don't care).

When this functionality is required, it can be ensured by programming the MEMHOLD value to meet the t_{WB} timing. However any CPU read access to the NAND Flash memory has a hold delay of (MEMHOLD + 2) HCLK cycles and CPU write access has a hold delay of (MEMHOLD) HCLK cycles inserted between the rising edge of the NWE signal and the next access.

To cope with this timing constraint, the attribute memory space can be used by programming its timing register with an ATTHOLD value that meets the t_{WB} timing, and by keeping the MEMHOLD value at its minimum value. The CPU must then use the common memory space for all NAND Flash read and write accesses, except when writing the last address byte to the NAND Flash device, where the CPU must write to the attribute memory space.

19.7.6 Computation of the error correction code (ECC) in NAND Flash memory

The FMC NAND Card controller includes two error correction code computation hardware blocks, one per memory bank. They reduce the host CPU workload when processing the ECC by software.

These two ECC blocks are identical and associated with Bank 2 and Bank 3. As a consequence, no hardware ECC computation is available for memories connected to Bank 4.

The ECC algorithm implemented in the FMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1 024, 2 048, 4 096 or 8 192 bytes read or written from/to the NAND Flash memory. It is based on the Hamming coding algorithm and consists in calculating the row and column parity.

The ECC modules monitor the NAND Flash data bus and read/write signals (NCE and NWE) each time the NAND Flash memory bank is active.

The ECC operates as follows:

- When accessing NAND Flash memory bank 2 or bank 3, the data present on the D[15:0] bus is latched and used for ECC computation.
- When accessing any other address in NAND Flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND Flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND Flash memory by the host CPU, the FMC_ECCR registers must be read to retrieve the computed value. Once read, they should be cleared by resetting the ECCEN bit to ‘0’. To compute a new data block, the ECCEN bit must be set to one in the FMC_PCR registers.

To perform an ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register.
2. Write data to the NAND Flash memory page. While the NAND page is written, the ECC block computes the ECC value.
3. Read the ECC value available in the FMC_ECCR register and store it in a variable.
4. Clear the ECCEN bit and then enable it in the FMC_PCR register before reading back the written data from the NAND page. While the NAND page is read, the ECC block computes the ECC value.
5. Read the new ECC value available in the FMC_ECCR register.
6. If the two ECC values are the same, no correction is required, otherwise there is an ECC error and the software correction routine returns information on whether the error can be corrected or not.

19.7.7 NAND Flash controller registers

NAND Flash control registers (FMC_PCR)

Address offset: 0x80

Reset value: 0x0000 0018

31	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.	ECCPS[2:0]			TAR3
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR[2:0]			TCLR[3:0]				Res.	Res.	ECCEN	PWID[1:0]		PTYP	PBKEN	PWAITEN	Res.
rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

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

Bits 19:17 **ECCPS[2:0]:** ECC page size

Defines the page size for the extended ECC:
000: 256 bytes
001: 512 bytes
010: 1024 bytes
011: 2048 bytes
100: 4096 bytes
101: 8192 bytes

Bits 16:13 **TAR[3:0]:** ALE to RE delay

Sets time from ALE low to RE low in number of AHB clock cycles (HCLK).
Time is: t_ar = (TAR + SET + 2) × THCLK where THCLK is the HCLK clock period
0000: 1 HCLK cycle (default)
1111: 16 HCLK cycles

Note: SET is MEMSET or ATTSET according to the addressed space.

Bits 12:9 **TCLR[3:0]:** CLE to RE delay

Sets time from CLE low to RE low in number of AHB clock cycles (HCLK).
Time is t_clr = (TCLR + SET + 2) × THCLK where THCLK is the HCLK clock period
0000: 1 HCLK cycle (default)
1111: 16 HCLK cycles

Note: SET is MEMSET or ATTSET according to the addressed space.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **ECCEN:** ECC computation logic enable bit

0: ECC logic is disabled and reset (default after reset),
1: ECC logic is enabled.

Bits 5:4 **PWID[1:0]:** Data bus width

Defines the external memory device width.
00: 8 bits
01: 16 bits (default after reset).
10: reserved.
11: reserved.

Bit 3 **PTYP:** Memory type

Defines the type of device attached to the corresponding memory bank:
0: Reserved, must be kept at reset value
1: NAND Flash (default after reset)

Bit 2 **PBKEN:** NAND Flash memory bank enable bit

Enables the memory bank. Accessing a disabled memory bank causes an ERROR on AHB bus
0: Corresponding memory bank is disabled (default after reset)
1: Corresponding memory bank is enabled

Bit 1 **PWAITEN:** Wait feature enable bit

Enables the Wait feature for the NAND Flash memory bank:
0: disabled
1: enabled

Bit 0 Reserved, must be kept at reset value.

FIFO status and interrupt register (FMC_SR)

Address offset: 0x84

Reset value: 0x0000 0040

This register contains information about the FIFO status and interrupt. The FMC features a FIFO that is used when writing to memories to transfer up to 16 words of data from the AHB.

This is used to quickly write to the FIFO and free the AHB for transactions to peripherals other than the FMC, while the FMC is draining its FIFO into the memory. One of these register bits indicates the status of the FIFO, for ECC purposes.

The ECC is calculated while the data are written to the memory. To read the correct ECC, the software must consequently wait until the FIFO is empty.

31	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.	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
									r	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **FEMPT:** FIFO empty
Read-only bit that provides the status of the FIFO
0: FIFO not empty
1: FIFO empty

Bit 5 **IFEN:** Interrupt falling edge detection enable bit
0: Interrupt falling edge detection request disabled
1: Interrupt falling edge detection request enabled

Bit 4 **ILEN:** Interrupt high-level detection enable bit
0: Interrupt high-level detection request disabled
1: Interrupt high-level detection request enabled

Bit 3 **IREN:** Interrupt rising edge detection enable bit
0: Interrupt rising edge detection request disabled
1: Interrupt rising edge detection request enabled

Bit 2 **IFS:** Interrupt falling edge status
The flag is set by hardware and reset by software.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred
Note: If this bit is written by software to 1 it is set.

Bit 1 **ILS:** Interrupt high-level status
The flag is set by hardware and reset by software.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0 **IRS:** Interrupt rising edge status

The flag is set by hardware and reset by software.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred

Note: If this bit is written by software to 1 it is set.

Common memory space timing register (FMC_PMEM)

Address offset: Address: 0x88

Reset value: 0xFCFC FCFC

The FMC_PMEM read/write register contains the timing information for NAND Flash memory bank. This information is used to access either the common memory space of the NAND Flash for command, address write access and data read/write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEMHIZ[7:0]								MEMHOLD[7:0]							
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
MEMWAIT[7:0]								MEMSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **MEMHIZ[7:0]:** Common memory x data bus Hi-Z time

Defines the number of HCLK clock cycles during which the data bus is kept Hi-Z after the start of a NAND Flash write access to common memory space on socket. This is only valid for write transactions:
0000 0000: 1 HCLK cycle
1111 1110: 255 HCLK cycles
1111 1111: reserved.

Bits 23:16 **MEMHOLD[7:0]:** Common memory hold time

Defines the number of HCLK clock cycles for write access and HCLK (+2) clock cycles for read access during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), for NAND Flash read or write access to common memory space on socket x:
0000 0000: reserved.
0000 0001: 1 HCLK cycle for write access / 3 HCLK cycles for read access
1111 1110: 254 HCLK cycles for write access / 256 HCLK cycles for read access
1111 1111: reserved.

Bits 15:8 **MEMWAIT[7:0]:** Common memory wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for NAND Flash read or write access to common memory space on socket. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:
0000 0000: reserved
0000 0001: 2HCLK cycles (+ wait cycle introduced by deasserting NWAIT)
1111 1110: 255 HCLK cycles (+ wait cycle introduced by deasserting NWAIT)
1111 1111: reserved.

Bits 7:0 **MEMSET[7:0]:** Common memory x setup time

Defines the number of HCLK (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for NAND Flash read or write access to common memory space on socket x:

0000 0000: 1 HCLK cycle
1111 1110: 255 HCLK cycles
1111 1111: reserved

Attribute memory space timing register (FMC_PATT)

Address offset: 0x8C

Reset value: 0xFCFC FCFC

The FMC_PATT read/write register contains the timing information for NAND Flash memory bank. It is used for 8-bit accesses to the attribute memory space of the NAND Flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to *Section 19.7.5: NAND Flash prewait functionality*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATTHIZ[7:0]								ATTHOLD[7:0]							
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
ATTWAIT[7:0]								ATTSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ATTHIZ[7:0]:** Attribute memory data bus Hi-Z time

Defines the number of HCLK clock cycles during which the data bus is kept in Hi-Z after the start of a NAND Flash write access to attribute memory space on socket. Only valid for writ transaction:

0000 0000: 0 HCLK cycle
1111 1110: 255 HCLK cycles
1111 1111: reserved.

Bits 23:16 **ATTHOLD[7:0]:** Attribute memory hold time

Defines the number of HCLK clock cycles for write access and HCLK (+2) clock cycles for read access during which the address is held (and data for write access) after the command deassertion (NWE, NOE), for NAND Flash read or write access to attribute memory space on socket:

0000 0000: reserved
0000 0001: 1 HCLK cycle for write access / 3 HCLK cycles for read access
1111 1110: 254 HCLK cycles for write access / 256 HCLK cycles for read access
1111 1111: reserved.

Bits 15:8 **ATTWAIT[7:0]:** Attribute memory wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for NAND Flash read or write access to attribute memory space on socket x. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:

0000 0000: reserved
0000 0001: 2 HCLK cycles (+ wait cycle introduced by deassertion of NWAIT)
1111 1110: 255 HCLK cycles (+ wait cycle introduced by deasserting NWAIT)
1111 1111: reserved.

Bits 7:0 **ATTSET[7:0]:** Attribute memory setup time

Defines the number of HCLK (+1) clock cycles to set up address before the command assertion (NWE, NOE), for NAND Flash read or write access to attribute memory space on socket:

0000 0000: 1 HCLK cycle

1111 1110: 255 HCLK cycles

1111 1111: reserved.

ECC result registers (FMC_ECCR)

Address offset: 0x94

Reset value: 0x0000 0000

This register contain the current error correction code value computed by the ECC computation modules of the FMC NAND controller. When the CPU reads the data from a NAND Flash memory page at the correct address (refer to *Section 19.7.6: Computation of the error correction code (ECC) in NAND Flash memory*), the data read/written from/to the NAND Flash memory are processed automatically by the ECC computation module. When X bytes have been read (according to the ECCPS field in the FMC_PCR registers), the CPU must read the computed ECC value from the FMC_ECC registers. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it otherwise. The FMC_ECCR register should be cleared after being read by setting the ECCEN bit to 0. To compute a new data block, the ECCEN bit must be set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC[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
ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ECC[31:0]:** ECC result

This field contains the value computed by the ECC computation logic. *Table 153* describes the contents of these bitfields.

Table 153. ECC result relevant bits

ECCPS[2:0]	Page size in bytes	ECC bits
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

19.7.8 FMC register map

Table 154. FMC 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	**FMC_BCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		WFDIS	CCLKEN	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0	0	0	0	0	0	0	0	0	1	1	0		0	0		1	0	1	1	0	1	1
0x08	**FMC_BCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		Res.	Res.	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0			0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x10	**FMC_BCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		Res.	Res.	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0			0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x18	**FMC_BCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		Res.	Res.	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0			0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x04	**FMC_BTR1**	DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	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	1	1
0x0C	**FMC_BTR2**	DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	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	1	1
0x14	**FMC_BTR3**	DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	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	1	1
0x1C	**FMC_BTR4**	DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	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	1	1
0x20	**FMC_PCSCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTB4EN	CNTB3EN	CNTB2EN	CNTB1EN	CSCOUNT[15:0]															
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	**FMC_BWTR1**	DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 154. FMC 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
0x10C	FMC_BWTR2	DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x114	FMC_BWTR3	DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x11C	FMC_BWTR4	DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x80	FMC_PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCPS [2:0]			TAR[3:0]				TCLR[3:0]				Res.	Res.	ECCEN	PWID [1:0]		PTYP	PBKEN	PWAITEN	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0			0	0	1	1	0	0	
0x84	FMC_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.	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
	Reset value																										1	0	0	0	0	0	0
0x88	FMC_PMEM	MEMHIZx[7:0]								MEMHOLDx[7:0]								MEMWAITx[7:0]								MEMSETx[7:0]							
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
0x8C	FMC_PATT	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
0x94	FMC_ECCR	ECCx[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 2.2 on page 81* for the register boundary addresses.

20 Quad-SPI interface (QUADSPI)

20.1 Introduction

The QUADSPI is a specialized communication interface targeting single, dual or quad SPI Flash memories. It can operate in any of the three following modes:

- indirect mode: all the operations are performed using the QUADSPI registers
- status polling mode: the external Flash memory status register is periodically read and an interrupt can be generated in case of flag setting
- memory-mapped mode: the external Flash memory is mapped to the device address space and is seen by the system as if it was an internal memory

Both throughput and capacity can be increased two-fold using dual-flash mode, where two Quad-SPI Flash memories are accessed simultaneously.

20.2 QUADSPI main features

- Three functional modes: indirect, status-polling, and memory-mapped
- Dual-flash mode, where 8 bits can be sent/received simultaneously by accessing two Flash memories in parallel.
- SDR and DDR support
- Fully programmable opcode for both indirect and memory mapped mode
- Fully programmable frame format for both indirect and memory mapped mode
- Integrated FIFO for reception and transmission
- 8, 16, and 32-bit data accesses are allowed
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error

20.3 QUADSPI functional description

20.3.1 QUADSPI block diagram

Figure 74. QUADSPI block diagram when dual-flash mode is disabled

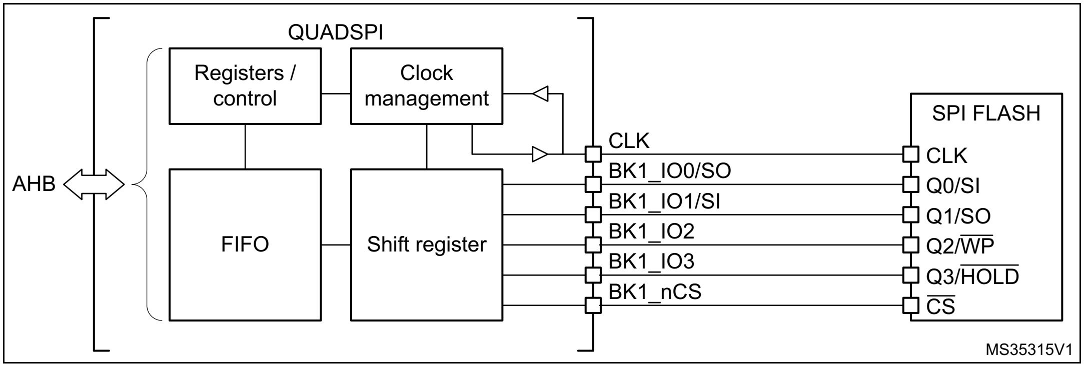
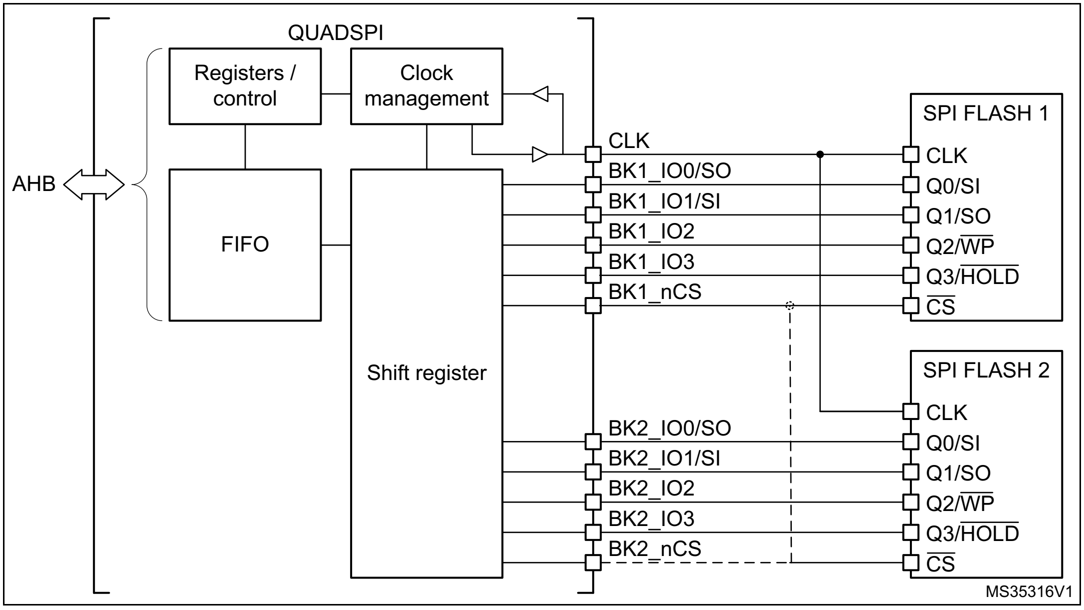


Figure 75. QUADSPI block diagram when dual-flash mode is enabled



20.3.2 QUADSPI pins

Table 155 lists the QUADSPI pins, six for interfacing with a single Flash memory, or 10 to 11 for interfacing with two Flash memories (FLASH 1 and FLASH 2) in dual-flash mode.

Table 155. QUADSPI pins

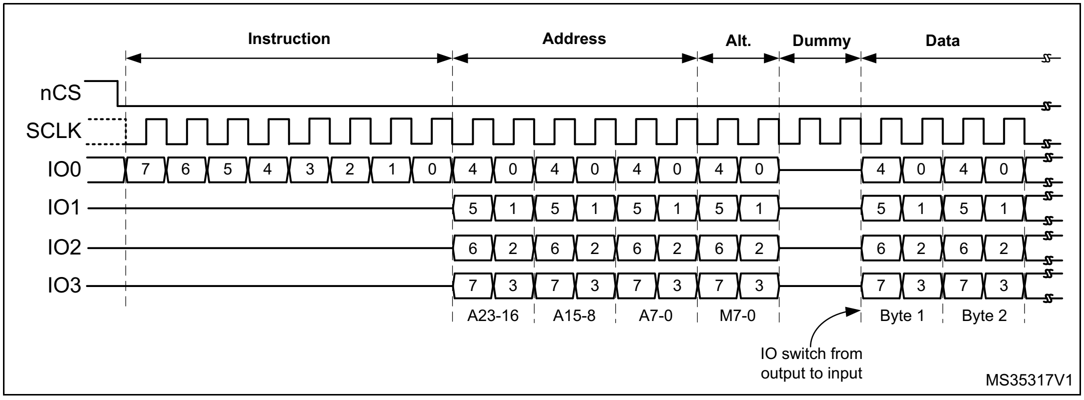
Signal name	Signal type	Description
CLK	Digital output	Clock to FLASH 1 and FLASH 2
BK1_IO0/SO	Digital input/output	Bidirectional IO in dual/quad modes or serial output in single mode, for FLASH 1
BK1_IO1/SI	Digital input/output	Bidirectional IO in dual/quad modes or serial input in single mode, for FLASH 1
BK1_IO2	Digital input/output	Bidirectional IO in quad mode, for FLASH 1
BK1_IO3	Digital input/output	Bidirectional IO in quad mode, for FLASH 1
BK2_IO0/SO	Digital input/output	Bidirectional IO in dual/quad modes or serial output in single mode, for FLASH 2
BK2_IO1/SI	Digital input/output	Bidirectional IO in dual/quad modes or serial input in single mode, for FLASH 2
BK2_IO2	Digital input/output	Bidirectional IO in quad mode, for FLASH 2
BK2_IO3	Digital input/output	Bidirectional IO in quad mode, for FLASH 2
BK1_nCS	Digital output	Chip select (active low) for FLASH 1. Can also be used for FLASH 2 if QUADSPI is always used in dual-flash mode.
BK2_nCS	Digital output	Chip select (active low) for FLASH 2. Can also be used for FLASH 1 if QUADSPI is always used in dual-flash mode.

20.3.3 QUADSPI command sequence

The QUADSPI communicates with the Flash memory using commands. Each command can include 5 phases: instruction, address, alternate byte, dummy, data. Any of these phases can be configured to be skipped, but at least one of the instruction, address, alternate byte, or data phase must be present.

nCS falls before the start of each command and rises again after each command finishes.

Figure 76. An example of a read command in quad mode



Instruction phase

During this phase, an 8-bit instruction, configured in INSTRUCTION field of QUADSPI_CCR[7:0] register, is sent to the Flash memory, specifying the type of operation to be performed.

Though most Flash memories can receive instructions only one bit at a time from the IO0/SO signal (single SPI mode), the instruction phase can optionally send 2 bits at a time (over IO0/IO1 in dual SPI mode) or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad SPI mode). This can be configured using the IMODE[1:0] field of QUADSPI_CCR[9:8] register.

When IMODE = 00, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

Address phase

In the address phase, 1-4 bytes are sent to the Flash memory to indicate the address of the operation. The number of address bytes to be sent is configured in the ADSIZE[1:0] field of QUADSPI_CCR[13:12] register. In indirect and automatic-polling modes, the address bytes to be sent are specified in the ADDRESS[31:0] field of QUADSPI_AR register, while in memory-mapped mode the address is given directly via the AHB (from the Cortex® or from a DMA).

The address phase can send 1 bit at a time (over SO in single SPI mode), 2 bits at a time (over IO0/IO1 in dual SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad SPI mode). This can be configured using the ADMODE[1:0] field of QUADSPI_CCR[11:10] register.

When ADMODE = 00, the address phase is skipped, and the command sequence proceeds directly to the next phase, if any.

Alternate-bytes phase

In the alternate-bytes phase, 1-4 bytes are sent to the Flash memory, generally to control the mode of operation. The number of alternate bytes to be sent is configured in the ABSIZE[1:0] field of QUADSPI_CCR[17:16] register. The bytes to be sent are specified in the QUADSPI_ABR register.

The alternate-bytes phase can send 1 bit at a time (over SO in single SPI mode), 2 bits at a time (over IO0/IO1 in dual SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad SPI mode). This can be configured using the ABMODE[1:0] field of QUADSPI_CCR[15:14] register.

When ABMODE = 00, the alternate-bytes phase is skipped, and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when dual-mode is used and only two cycles are used for the alternate bytes. In this case, firmware can use quad-mode (ABMODE = 11) and send a byte with bits 7 and 3 of ALTERNATE set to ‘1’ (keeping the IO3 line high), and bits 6 and 2 set to ‘0’ (keeping the IO2 line low). In this case the upper two bits of the nibble to be sent are placed in bits 4:3 of ALTERNATE while the lower two bits are placed in bits 1 and 0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE should be set to 0x8A (1000_1010).

Dummy-cycles phase

In the dummy-cycles phase, 1-31 cycles are given without any data being sent or received, in order to allow the Flash memory the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified in the DCYC[4:0] field of QUADSPI_CCR[22:18] register. In both SDR and DDR modes, the duration is specified as a number of full CLK cycles.

When DCYC is zero, the dummy-cycles phase is skipped, and the command sequence proceeds directly to the data phase, if present.

The operating mode of the dummy-cycles phase is determined by DMODE.

In order to assure enough “turn-around” time for changing the data signals from output mode to input mode, there must be at least one dummy cycle when using dual or quad mode to receive data from the Flash memory.

Data phase

During the data phase, any number of bytes can be sent to, or received from the Flash memory.

In indirect and automatic-polling modes, the number of bytes to be sent/received is specified in the QUADSPI_DLR register.

In indirect write mode the data to be sent to the Flash memory must be written to the QUADSPI_DR register, while in indirect read mode the data received from the Flash memory is obtained by reading from the QUADSPI_DR register.

In memory-mapped mode, the data which is read is sent back directly over the AHB to the Cortex or to a DMA.

The data phase can send/receive 1 bit at a time (over SO/SI in single SPI mode), 2 bits at a time (over IO0/IO1 in dual SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad SPI mode). This can be configured using the ABMODE[1:0] field of QUADSPI_CCR[15:14] register.

When DMODE = 00, the data phase is skipped, and the command sequence finishes immediately by raising nCS. This configuration must only be used in only indirect write mode.

20.3.4 QUADSPI signal interface protocol modes

Single SPI mode

Legacy SPI mode allows just a single bit to be sent/received serially. In this mode, data is sent to the Flash memory over the SO signal (whose I/O shared with IO0). Data received from the Flash memory arrives via SI (whose I/O shared with IO1).

The different phases can each be configured separately to use this single bit mode by setting the IMODE/ADMODE/ABMODE/DMODE fields (in QUADSPI_CCR) to 01.

In each phase which is configured in single mode:

- IO0 (SO) is in output mode
- IO1 (SI) is in input mode (high impedance)
- IO2 is in output mode and forced to ‘0’
- IO3 is in output mode and forced to ‘1’ (to deactivate the “hold” function)

This is the case even for the dummy phase if DMODE = 01.

Dual SPI mode

In dual SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI_CCR register to 10.

In each phase which is configured in dual mode:

- IO0/IO1 are at high-impedance (input) during the data phase for read operations, and outputs in all other cases
- IO2 is in output mode and forced to ‘0’
- IO3 is in output mode and forced to ‘1’

In the dummy phase when DMODE = 01, IO0/IO1 are always high-impedance.

Quad SPI mode

In quad SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use quad SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI_CCR register to 11.

In each phase which is configured in quad mode, IO0/IO1/IO2/IO3 are all are at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE = 11, IO0/IO1/IO2/IO3 are all high-impedance.

IO2 and IO3 are used only in Quad SPI mode. If none of the phases are configured to use Quad SPI mode, then the pins corresponding to IO2 and IO3 can be used for other functions even while QUADSPI is active.

SDR mode

By default, the DDRM bit (QUADSPI_CCR[31]) is 0 and the QUADSPI operates in single data rate (SDR) mode.

In SDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the QUADSPI assumes that the Flash memories also send the data using CLK's falling edge. By default (when SSHIFT = 0), the signals are sampled using the following (rising) edge of CLK.

DDR mode

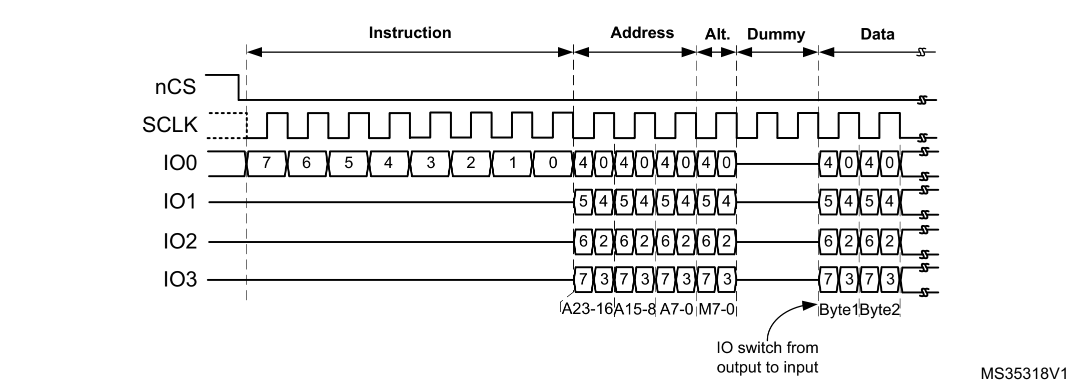
When the DDRM bit (QUADSPI_CCR[31]) is set to 1, the QUADSPI operates in double data rate (DDR) mode.

In DDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals in the address/alternate-byte/data phases, a bit is sent on each of the falling and rising edges of CLK.

The instruction phase is not affected by DDRM. The instruction is always sent using CLK's falling edge.

When receiving data in DDR mode, the QUADSPI assumes that the Flash memories also send the data using both rising and falling CLK edges. When DDRM = 1, firmware must clear SSHIFT bit (bit 4 of QUADSPI_CR). Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

Figure 77. An example of a DDR command in quad mode



Dual-flash mode

When the DFM bit (bit 6 of QUADSPI_CR) is 1, the QUADSPI is in dual-flash mode, where two external quad SPI Flash memories (FLASH 1 and FLASH 2) are used in order to send/receive 8 bits (or 16 bits in DDR mode) every cycle, effectively doubling the throughput as well as the capacity.

Each of the Flash memories use the same CLK and optionally the same nCS signals, but each have separate IO0, IO1, IO2, and IO3 signals.

Dual-flash mode can be used in conjunction with single-bit, dual-bit, and quad-bit modes, as well as with either SDR or DDR mode.

The Flash memory size, as specified in FSIZE[4:0] (QUADSPI_DCR[20:16]), should reflect the total Flash memory capacity, which is double the size of one individual component.

If address X is even, then the byte which the QUADSPI gives for address X is the byte at the address X/2 of FLASH 1, and the byte which the QUADSPI gives for address X+1 is the byte at the address X/2 of FLASH 2. In other words, bytes at even addresses are all stored in FLASH 1 and bytes at odd addresses are all stored in FLASH 2.

When reading the Flash memories status registers in dual-flash mode, twice as many bytes should be read compared to doing the same read in single-flash mode. This means that if each Flash memory gives 8 valid bits after the instruction for fetching the status register, then the QUADSPI must be configured with a data length of 2 bytes (16 bits), and the QUADSPI receives one byte from each Flash memory. If each Flash memory gives a status of 16 bits, then the QUADSPI must be configured to read 4 bytes to get all the status bits of both Flash memories in dual-flash mode. The least-significant byte of the result (in the data register) is the least-significant byte of FLASH 1 status register, while the next byte is the least-significant byte of FLASH 2 status register. Then, the third byte of the data register is FLASH 1 second byte, while the forth byte is FLASH 2 second byte (in the case that the Flash memories have 16-bit status registers).

An even number of bytes must always be accessed in dual-flash mode. For this reason, bit 0 of the data length field (QUADSPI_DLR[0]) is stuck at 1 when DRM = 1.

In dual-flash mode, the behavior of FLASH 1 interface signals are basically the same as in normal mode. FLASH 2 interface signals have exactly the same waveforms as FLASH 1 during the instruction, address, alternate-byte, and dummy-cycles phases. In other words, each Flash memory always receives the same instruction and the same address. Then, during the data phase, the BK1_IOx and BK2_IOx buses are both transferring data in parallel, but the data that are sent to (or received from) FLASH 1 are distinct from those of FLASH 2.

20.3.5 QUADSPI indirect mode

When in indirect mode, commands are started by writing to QUADSPI registers and data is transferred by writing or reading the data register, in the same way as for other communication peripherals.

When FMODE = 00 (QUADSPI_CCR[27:26]), the QUADSPI is in indirect write mode, where bytes are sent to the Flash memory during the data phase. Data are provided by writing to the data register (QUADSPI_DR).

When FMODE = 01, the QUADSPI is in indirect read mode, where bytes are received from the Flash memory during the data phase. Data are recovered by reading QUADSPI_DR.

The number of bytes to be read/written is specified in the data length register (QUADSPI_DLR). If QUADSPI_DLR = 0xFFFF_FFFF (all 1’s), then the data length is considered undefined and the QUADSPI simply continues to transfer data until the end of Flash memory (as defined by FSIZE) is reached. If no bytes are to be transferred, DMODE (QUADSPI_CCR[25:24]) should be set to 00.

If QUADSPI_DLR = 0xFFFF_FFFF and FSIZE = 0x1F (max value indicating a 4GB Flash memory), then in this special case the transfers continue indefinitely, stopping only after an abort request or after the QUADSPI is disabled. After the last memory address is read (at address 0xFFFF_FFFF), reading continues with address = 0x0000_0000.

When the programmed number of bytes to be transmitted or received is reached, TCF is set and an interrupt is generated if TCIE = 1. In the case of undefined number of data, the TCF is set when the limit of the external SPI memory is reached according to the Flash memory size defined in the QUADSPI_CR.

Triggering the start of a command

Essentially, a command starts as soon as firmware gives the last information that is necessary for this command. Depending on the QUADSPI configuration, there are three different ways to trigger the start of a command in indirect mode. The commands starts immediately after:

1. a write is performed to INSTRUCTION[7:0] (QUADSPI_CCR), if no address is necessary (when ADMODE = 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
2. a write is performed to ADDRESS[31:0] (QUADSPI_AR), if an address is necessary (when ADMODE != 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
3. a write is performed to DATA[31:0] (QUADSPI_DR), if an address is necessary (when ADMODE != 00) and if data needs to be provided by the firmware (when FMODE = 00 and DMODE != 00)

Writes to the alternate byte register (QUADSPI_ABR) never trigger the communication start. If alternate bytes are required, they must be programmed before.

As soon as a command is started, the BUSY bit (bit 5 of QUADSPI_SR) is automatically set.

FIFO and data management

In indirect mode, data go through a 16-byte FIFO which is internal to the QUADSPI. FLEVEL[4:0] (QUADSPI_SR[12:8]) indicates how many bytes are currently being held in the FIFO.

In indirect write mode (FMODE = 00), firmware adds data to the FIFO when it writes QUADSPI_DR. Word writes add 4 bytes to the FIFO, halfword writes add 2 bytes, and byte writes add only 1 byte. If firmware adds too many bytes to the FIFO (more than is indicated by DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

Byte/halfword accesses to QUADSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES[3:0] is used to define a FIFO threshold. When the threshold is reached, the FTF (FIFO threshold flag) is set. In indirect read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there are data in the FIFO after the last byte is read from the Flash memory, regardless of the FTHRES setting. In indirect write mode, FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when FTF is set. If DMAEN = 1, a DMA transfer is initiated when FTF is set. FTF is cleared by HW as soon as the threshold condition is no longer true (after enough data is transferred by the CPU or DMA).

20.3.6 QUADSPI status flag polling mode

In automatic-polling mode, the QUADSPI periodically starts a command to read a defined number of status bytes (up to 4). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value.

The accesses to the Flash memory begin in the same way as in indirect read mode: if no address is required (AMODE = 00), accesses begin as soon as the QUADSPI_CCR is written. Otherwise, if an address is required, the first access begins when QUADSPI_AR is written. BUSY goes high at this point and stays high even between the periodic accesses.

The contents of MASK[31:0] (QUADSPI_PSMAR) are used to mask the data from the Flash memory in automatic-polling mode. If the MASK[n] = 0, then bit n of the result is masked and not considered. If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] (QUADSPI_PSMAR), then there is a match for bit n.

If the polling match mode bit (PMM, bit 23 of QUADSPI_CR) is 0, then “AND” match mode is activated. This means status match flag (SMF) is set only when there is a match on all of the unmasked bits.

If PMM = 1, then “OR” match mode is activated. This means SMF is set if there is a match on any of the unmasked bits.

An interrupt is called when SMF is set if SMIE = 1.

If the automatic-polling-mode-stop (APMS) bit is set, operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at ‘1’ and the periodic accesses continue until there is an abort or the QUADSPI is disabled (EN = 0).

The data register (QUADSPI_DR) contains the latest received status bytes (the FIFO is deactivated). The content of the data register is not affected by the masking used in the matching logic. The FTF status bit is set as soon as a new reading of the status is complete, and FTF is cleared as soon as the data is read.

20.3.7 QUADSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

It is forbidden to access QUADSPI Flash bank area before having properly configured and enabled the QUADSPI peripheral.

No more than 256MB can addressed even if the Flash memory capacity is larger.

If an access is made to an address outside of the range defined by FSIZE but still within the 256MB range, then a bus error is given. The effect of this error depends on the bus master that attempted the access:

- If it is the Cortex® CPU, bus fault exception is generated when enabled (or a hard fault exception when bus fault is disabled)
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

Byte, halfword, and word access types are all supported.

Support for execute in place (XIP) operation is implemented, where the QUADSPI anticipates the next access and load in advance the byte at the following address. If the subsequent access is indeed made at a continuous address, the access is completed faster since the value is already prefetched.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with nCS maintained low, even if no access to the Flash memory occurs for a long time. Since Flash memories tend to consume more when nCS is held low, the application might want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that nCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without any access since when the FIFO becomes full with prefetch data.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is a timeout, there is an abort, or the peripheral is disabled.

20.3.8 QUADSPI Flash memory configuration

The device configuration register (QUADSPI_DCR) can be used to specify the characteristics of the external SPI Flash memory.

The FSIZE[4:0] field defines the size of external memory using the following formula:

Number of bytes in Flash memory = $2^{[FSIZE+1]}$

FSIZE+1 is effectively the number of address bits required to address the Flash memory. The Flash memory capacity can be up to 4GB (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256MB.

If DFM = 1, FSIZE indicates the total capacity of the two Flash memories together.

When the QUADSPI executes two commands, one immediately after the other, it raises the chip select signal (nCS) high between the two commands for only one CLK cycle by default. If the Flash memory requires more time between commands, the chip select high time (CSHT) field can be used to specify the minimum number of CLK cycles (up to 8) that nCS must remain high.

The clock mode (CKMODE) bit indicates the CLK signal logic level in between commands (when nCS = 1).

20.3.9 QUADSPI delayed data sampling

By default, the QUADSPI samples the data driven by the Flash memory one half of a CLK cycle after the Flash memory drives the signal.

In case of external signal delays, it may be beneficial to sample the data later. Using the SSHIFT bit (bit 4 of QUADSPI_CR), the sampling of the data can be shifted by half of a CLK cycle.

Clock shifting is not supported in DDR mode: the SSHIFT bit must be clear when DDRM bit is set.

20.3.10 QUADSPI configuration

The QUADSPI configuration is done in two phases:

- QUADSPI IP configuration
- QUADSPI Flash memory configuration

Once configured and enabled, the QUADSPI can be used in one of its three operating modes: indirect mode, status-polling mode, or memory-mapped mode.

QUADSPI configuration

The QUADSPI is configured using the QUADSPI_CR. The user shall configure the clock prescaler division factor and the sample shifting settings for the incoming data.

DDR mode can be set through the DDRM bit. When setting QUADSPI interface in DDR mode, the internal divider of kernel clock must be set with a division ratio of 2 or more. Once enabled, the address and the alternate bytes are sent on both clock edges and the data are sent/received on both clock edges. Regardless of the DDRM bit setting, instructions are always sent in SDR mode.

The DMA requests are enabled setting the DMAEN bit. In case of interrupt usage, their respective enable bit can be also set during this phase.

FIFO level for either DMA request generation or interrupt generation is programmed in the FTHRES bits.

If timeout counter is needed, the TCEN bit can be set and the timeout value programmed in the QUADSPI_LPTR register.

Dual-flash mode can be activated by setting DFM to 1.

QUADSPI Flash memory configuration

The parameters related to the targeted external Flash memory are configured through the QUADSPI_DCR register.The user shall program the Flash memory size in the FSIZE bits, the Chip Select minimum high time in the CSHT bits, and the functional mode (Mode 0 or Mode 3) in the MODE bit.

20.3.11 QUADSPI usage

The operating mode is selected using FMODE[1:0] (QUADSPI_CCR[27:26]).

Indirect mode procedure

When FMODE is programmed to 00, indirect write mode is selected and data can be sent to the Flash memory. With FMODE = 01, indirect read mode is selected where data can be read from the Flash memory.

When the QUADSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in the QUADSPI_DLR.
2. Specify the frame format, mode and instruction code in the QUADSPI_CCR.
3. Specify optional alternate byte to be sent right after the address phase in the QUADSPI_ABR.
4. Specify the operating mode in the QUADSPI_CR. If FMODE = 00 (indirect write mode) and DMAEN = 1, then QUADSPI_AR should be specified before QUADSPI_CR, because otherwise QUADSPI_DR might be written by the DMA before QUADSPI_AR is updated (if the DMA controller has already been enabled)
5. Specify the targeted address in the QUADSPI_AR.
6. Read/Write the data from/to the FIFO through the QUADSPI_DR.

When writing the control register (QUADSPI_CR) the user specifies the following settings:

- The enable bit (EN) set to ‘1’
- The DMA enable bit (DMAEN) for transferring data to/from RAM
- Timeout counter enable bit (TCEN)
- Sample shift setting (SSHIFT)
- FIFO threshold level (FTRHES) to indicate when the FTF flag should be set
- Interrupt enables
- Automatic polling mode parameters: match mode and stop mode (valid when FMODE = 11)
- Clock prescaler

When writing the communication configuration register (QUADSPI_CCR) the user specifies the following parameters:

- The instruction byte through the INSTRUCTION bits
- The way the instruction has to be sent through the IMODE bits (1/2/4 lines)
- The way the address has to be sent through the ADMODE bits (None/1/2/4 lines)
- The address size (8/16/24/32-bit) through the ADSIZE bits
- The way the alternate bytes have to be sent through the ABMODE (None/1/2/4 lines)
- The alternate bytes number (1/2/3/4) through the ABSIZE bits
- The presence or not of dummy bytes through the DBMODE bit
- The number of dummy bytes through the DCYC bits
- The way the data have to be sent/received (None/1/2/4 lines) through the DMODE bits

If neither the address register (QUADSPI_AR) nor the data register (QUADSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as QUADSPI_CCR is written. This is the case when both ADMODE and DMODE are 00, or if just ADMODE = 00 when in indirect read mode (FMODE = 01).

When an address is required (ADMODE is not 00) and the data register does not need to be written (when FMODE = 01 or DMODE = 00), the command sequence starts as soon as the address is updated with a write to QUADSPI_AR.

In case of data transmission (FMODE = 00 and DMODE! = 00), the communication start is triggered by a write in the FIFO through QUADSPI_DR.

Status flag polling mode

The status flag polling mode is enabled setting the FMODE field (QUADSPI_CCR[27:26]) to 10. In this mode, the programmed frame is sent and the data retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in QUADSPI_DLR, it is ignored and only 4 bytes are read.

The periodicity is specified in the QUADSPI_PISR register.

Once the status data is retrieved, it can internally be processed i order to:

- set the status match flag and generate an interrupt if enabled
- stop automatically the periodic retrieving of the status bytes

The received value can be masked with the value stored in the QUADSPI_PSMKR and ORed or ANDed with the value stored in the QUADSPI_PSMAR.

In case of match, the status match flag is set and an interrupt is generated if enabled, and the QUADSPI can be automatically stopped if the AMPS bit is set.

In any case, the latest retrieved value is available in the QUADSPI_DR.

Memory-mapped mode

In memory-mapped mode, the external Flash memory is seen as internal memory but with some latency during accesses. Only read operations are allowed to the external Flash memory in this mode.

Memory-mapped mode is entered by setting the FMODE to 11 in the QUADSPI_CCR register.

The programmed instruction and frame is sent when a master is accessing the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate linear reads. Any access to QUADSPI_DR in this mode returns zero.

The data length register (QUADSPI_DLR) has no meaning in memory-mapped mode.

20.3.12 Sending the instruction only once

Some Flash memories (e.g. Winbound) might provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. One can take advantage of such a feature using the SIOO bit (QUADSPI_CCR[28]).

SIOO is valid for all functional modes (indirect, automatic polling, and memory-mapped). If the SIOO bit is set, the instruction is sent only for the first command following a write to QUADSPI_CCR. Subsequent command sequences skip the instruction phase, until there is a write to QUADSPI_CCR.

SIOO has no effect when IMODE = 00 (no instruction).

20.3.13 QUADSPI error management

An error can be generated in the following case:

- In indirect mode or status flag polling mode when a wrong address is programmed in the QUADSPI_AR (according to the Flash memory size defined by FSIZE[4:0] in the QUADSPI_DCR): this sets the TEF and an interrupt is generated if enabled.
- Also in indirect mode, if the address plus the data length exceeds the Flash memory size, TEF is set as soon as the access is triggered.
- In memory-mapped mode, when an out of range access is done by a master or when the QUADSPI is disabled: this generates a bus error as a response to the faulty bus master request.
- When a master is accessing the memory mapped space while the memory mapped mode is disabled: this generates a bus error as a response to the faulty bus master request.

20.3.14 QUADSPI busy bit and abort functionality

Once the QUADSPI starts an operation with the Flash memory, the BUSY bit is automatically set in the QUADSPI_SR.

In indirect mode, the BUSY bit is reset once the QUADSPI has completed the requested command sequence and the FIFO is empty.

In automatic-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1, or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on a timeout event or on an abort.

Any operation can be aborted by setting the ABORT bit in the QUADSPI_CR. Once the abort is completed, the BUSY bit and the ABORT bit are automatically reset, and the FIFO is flushed.

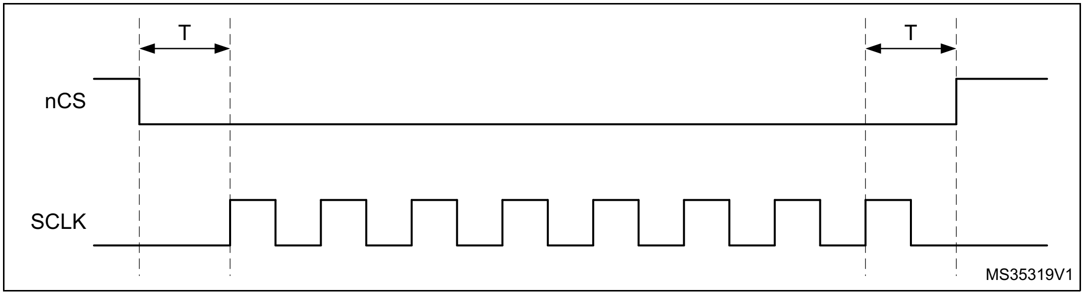
Note: Some Flash memories might misbehave if a write operation to a status registers is aborted.

20.3.15 nCS behavior

By default, nCS is high, deselecting the external Flash memory. nCS falls before an operation begins and rises as soon as it finishes.

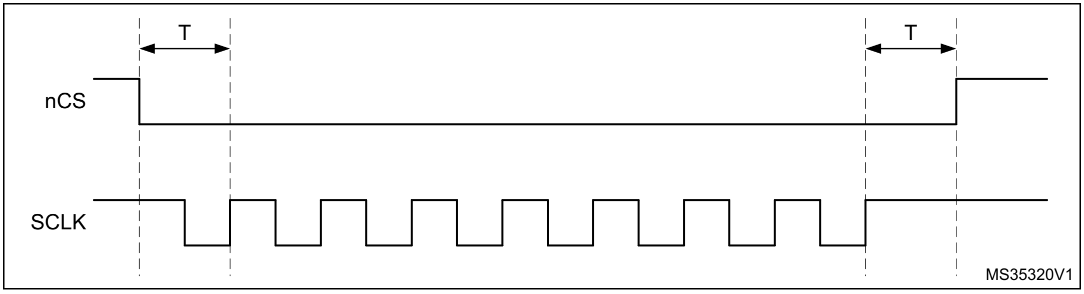
When CKMODE = 0 (“mode0”, where CLK stays low when no operation is in progress) nCS falls one CLK cycle before an operation first rising CLK edge, and nCS rises one CLK cycle after the operation final rising CLK edge, as shown in *Figure 78*.

Figure 78. nCS when CKMODE = 0 (T = CLK period)



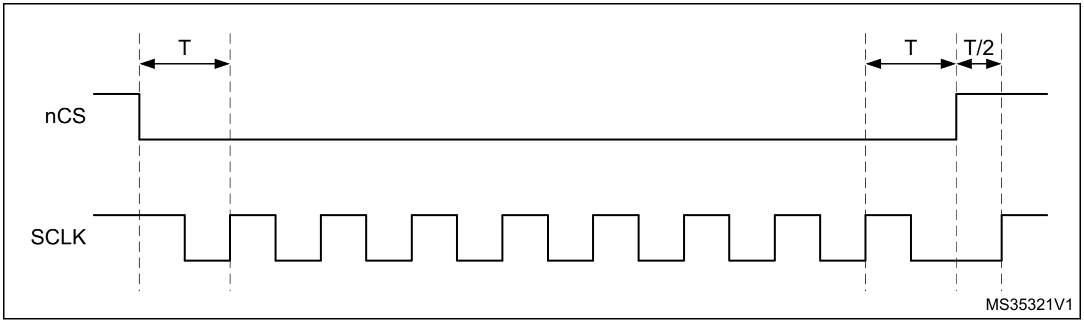
When CKMODE=1 (“mode3”, where CLK goes high when no operation is in progress) and DDRM=0 (SDR mode), nCS still falls one CLK cycle before an operation first rising CLK edge, and nCS rises one CLK cycle after the operation final rising CLK edge, as shown in *Figure 79*.

Figure 79. nCS when CKMODE = 1 in SDR mode (T = CLK period)



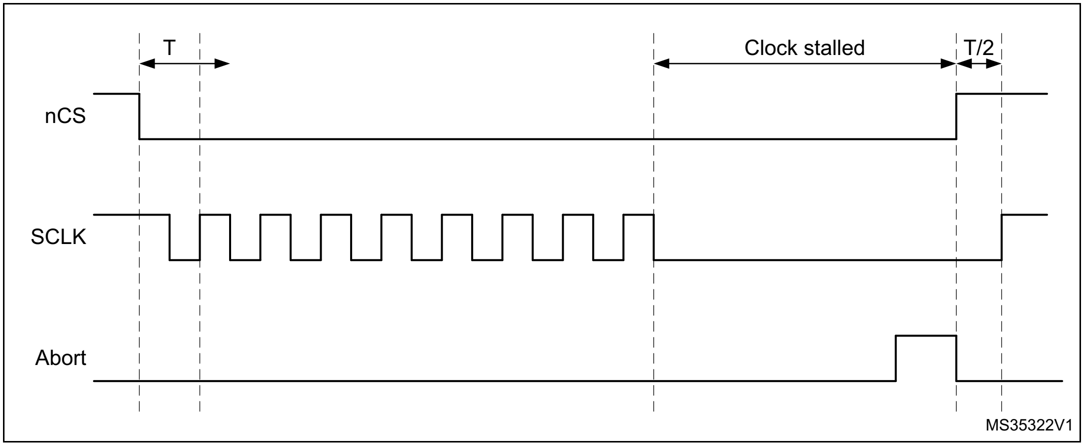
When CKMODE = 1 (“mode3”) and DDRM = 1 (DDR mode), nCS falls one CLK cycle before an operation first rising CLK edge, and nCS rises one CLK cycle after the operation final active rising CLK edge, as shown in *Figure 80*. Because DDR operations must finish with a falling edge, CLK is low when nCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Figure 80. nCS when CKMODE = 1 in DDR mode (T = CLK period)



When the FIFO stays full in a read operation or if the FIFO stays empty in a write operation, the operation stalls and CLK stays low until firmware services the FIFO. If an abort occurs when an operation is stalled, nCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later, as shown in *Figure 81*.

Figure 81. nCS when CKMODE = 1 with an abort (T = CLK period)



When not in dual-flash mode (DFM = 0) and FSEL = 0 (default value), only FLASH 1 is accessed and thus BK2_nCS stays high, if FSEL = 1, only FLASH 2 is accessed and BK1_nCS stays high. In dual-flash mode, BK2_nCS behaves exactly the same as BK1_nCS. Thus, if there is a FLASH 2 and if the application is dual-flash mode only, then BK1_nCS signal can be used for FLASH 2 as well, and the pin devoted to BK2_nCS can be used for other functions.

20.4 QUADSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

Table 156. QUADSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

20.5 QUADSPI registers

20.5.1 QUADSPI control register (QUADSPI_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESCALER[7:0]								PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
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.	FTHRES[3:0]				FSEL	DFM	Res.	SSHIFT	TCEN	DMAEN	ABORT	EN
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:24 **PRESCALER[7:0]**: Clock prescaler

This field defines the scaler factor for generating CLK based on the clock (value+1).

0: F_{CLK} = F, clock used directly as QUADSPI CLK (prescaler bypassed)

1: F_{CLK} = F/2

2: F_{CLK} = F/3

...

255: F_{CLK} = F/256

For odd clock division factors, CLK's duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high.

This field can be modified only when BUSY = 0.

When setting QUADSPI interface in DDR mode, the prescaler must be set with a division ratio of 2 or more.

Bit 23 **PMM**: Polling match mode

This bit indicates which method should be used for determining a “match” during automatic polling mode.

0: AND match mode. SMF is set if all the unmasked bits received from the Flash memory match the corresponding bits in the match register.

1: OR match mode. SMF is set if any one of the unmasked bits received from the Flash memory matches its corresponding bit in the match register.

This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic poll mode stop

This bit determines if automatic polling is stopped after a match.

0: Automatic polling mode is stopped only by abort or by disabling the QUADSPI.

1: Automatic polling mode stops as soon as there is a match.

This bit can be modified only when BUSY = 0.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: TimeOut interrupt enable

This bit enables the TimeOut interrupt.

0: Interrupt disable

1: Interrupt enabled

Bit 19 **SMIE**: Status match interrupt enable

This bit enables the status match interrupt.
0: Interrupt disable
1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable

This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable

This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable

This bit enables the transfer error interrupt.
0: Interrupt disable
1: Interrupt enabled

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **FTHRES[3:0]**: FIFO threshold level

Defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag (bit FTF in register QUADSPI_SR) to be set.
0: In indirect write mode (FMODE = 00) FTF is set if there are 1 or more free bytes location left in the FIFO or indirect read mode (FMODE = 01) FTF is set if there are 1 or more valid bytes that can be read from the FIFO
1: In indirect write mode (FMODE = 00) FTF is set if there are 2 or more free bytes location left in the FIFO or indirect read mode (FMODE = 01) FTF is set if there are 2 or more valid bytes that can be read from the FIFO
...
15: In indirect write mode (FMODE = 00) FTF is set if there are 16 free bytes location left in the FIFO or indirect read mode (FMODE = 01) FTF is set if there are 16 valid bytes that can be read from the FIFO
If DMAEN = 1, then the DMA controller for the corresponding channel must be disabled before changing the FTHRES value.

Bit 7 **FSEL**: Flash memory selection

This bit selects the Flash memory to be addressed in single flash mode (when DFM = 0).
0: FLASH 1 selected
1: FLASH 2 selected
This bit can be modified only when BUSY = 0.
This bit is ignored when DFM = 1.

Bit 6 **DFM**: Dual-flash mode

This bit activates dual-flash mode, where two external Flash memories are used simultaneously to double throughput and capacity.
0: Dual-flash mode disabled
1: Dual-flash mode enabled
This bit can be modified only when BUSY = 0.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **SSHIFT**: Sample shift

By default, the QUADSPI samples data 1/2 of a CLK cycle after the data is driven by the Flash memory. This bit allows the data to be sampled later in order to account for external signal delays.

0: No shift

1: 1/2 cycle shift

Firmware must assure that SSHIFT = 0 when in DDR mode (when DDRM = 1).

This field can be modified only when BUSY = 0.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when memory-mapped mode (FMODE = 11) is selected. Activating this bit causes the chip select (nCS) to be released (and thus reduces consumption) if there has not been an access after a certain amount of time, where this time is defined by TIMEOUT[15:0] (QUADSPI_LPTR).

Enable the timeout counter.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with nCS maintained low, even if no access to the Flash memory occurs for a long time. Since Flash memories tend to consume more when nCS is held low, the application might want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that nCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without an access since when the FIFO becomes full with prefetch data.

0: Timeout counter is disabled, and thus the chip select (nCS) remains active indefinitely after an access in memory-mapped mode.

1: Timeout counter is enabled, and thus the chip select is released in memory-mapped mode after TIMEOUT[15:0] cycles of Flash memory inactivity.

This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, DMA can be used to input or output data via the QUADSPI_DR register. DMA transfers are initiated when the FIFO threshold flag, FTF, is set.

0: DMA is disabled for indirect mode

1: DMA is enabled for indirect mode

Bit 1 **ABORT**: Abort request

This bit aborts the on-going command sequence. It is automatically reset once the abort is complete.

This bit stops the current transfer.

In polling mode or memory-mapped mode, this bit also reset the APM bit or the DM bit.

0: No abort requested

1: Abort requested

Bit 0 **EN**: Enable

Enable the QUADSPI.

0: QUADSPI is disabled

1: QUADSPI is enabled

20.5.2 QUADSPI device configuration register (QUADSPI_DCR)

Address offset: 0x0004

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.	FSIZE[4:0]				
											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.	CSHT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	CK MODE
					rw	rw	rw								rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **FSIZE[4:0]**: Flash memory size

This field defines the size of external memory using the following formula:

Number of bytes in Flash memory = $2^{[FSIZE+1]}$

FSIZE+1 is effectively the number of address bits required to address the Flash memory. The Flash memory capacity can be up to 4GB (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256MB.

If DFM = 1, FSIZE indicates the total capacity of the two Flash memories together.

This field can be modified only when BUSY = 0.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **CSHT[2:0]**: Chip select high time

CSHT+1 defines the minimum number of CLK cycles which the chip select (nCS) must remain high between commands issued to the Flash memory.

0: nCS stays high for at least 1 cycle between Flash memory commands

1: nCS stays high for at least 2 cycles between Flash memory commands

...

7: nCS stays high for at least 8 cycles between Flash memory commands

This field can be modified only when BUSY = 0.

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

Bit 0 **CKMODE**: Mode 0 / mode 3

This bit indicates the level that CLK takes between commands (when nCS = 1).

0: CLK must stay low while nCS is high (chip select released). This is referred to as mode 0.

1: CLK must stay high while nCS is high (chip select released). This is referred to as mode 3.

This field can be modified only when BUSY = 0.

20.5.3 QUADSPI status register (QUADSPI_SR)

Address offset: 0x0008

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.	FLEVEL[4:0]					Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **FLEVEL[4:0]**: FIFO level

This field gives the number of valid bytes which are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 16 when it is full. In memory-mapped mode and in automatic status polling mode, FLEVEL is zero.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **BUSY**: Busy

This bit is set when an operation is on going. This bit clears automatically when the operation with the Flash memory is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic polling mode when the unmasked received data matches the corresponding bits in the match register (QUADSPI_PSMAR). It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold is reached, or if there is any data left in the FIFO after reads from the Flash memory are complete. It is cleared automatically as soon as threshold condition is no longer true.

In automatic polling mode this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data is transferred or in any mode when the transfer is aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode. It is cleared by writing 1 to CTEF.

20.5.4 QUADSPI flag clear register (QUADSPI_FCR)

Address offset: 0x000C

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.	CTOF	CSMF	Res.	CTCF	CTEF
											w	w		w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag

Writing 1 clears the TOF flag in the QUADSPI_SR register

Bit 3 **CSMF**: Clear status match flag

Writing 1 clears the SMF flag in the QUADSPI_SR register

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag

Writing 1 clears the TCF flag in the QUADSPI_SR register

Bit 0 **CTEF**: Clear transfer error flag

Writing 1 clears the TEF flag in the QUADSPI_SR register

20.5.5 QUADSPI data length register (QUADSPI_DLR)

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DL[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
DL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect and status-polling modes. A value no greater than 3 (indicating 4 bytes) should be used for status-polling mode.

All 1s in indirect mode means undefined length, where QUADSPI continues until the end of memory, as defined by FSIZE.

0x0000_0000: 1 byte is to be transferred
0x0000_0001: 2 bytes are to be transferred
0x0000_0002: 3 bytes are to be transferred
0x0000_0003: 4 bytes are to be transferred
...
0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred
0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred
0xFFFF_FFFF: undefined length -- all bytes until the end of Flash memory (as defined by FSIZE) are to be transferred. Continue reading indefinitely if FSIZE = 0x1F.

DL[0] is stuck at '1' in dual-flash mode (DFM = 1) even when '0' is written to this bit, thus assuring that each access transfers an even number of bytes.

This field has no effect when in memory-mapped mode (FMODE = 10).

This field can be written only when BUSY = 0.

20.5.6 QUADSPI communication configuration register (QUADSPI_CCR)

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DDRM	DHHC	Res.	SIOO	FMODE[1:0]		DMODE[1:0]		Res.	DCYC[4:0]					ABSIZE[1:0]	
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
ABMODE[1:0]		ADSIZE[1:0]		ADMODE[1:0]		IMODE[1:0]		INSTRUCTION[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DDRM**: Double data rate mode

This bit sets the DDR mode for the address, alternate byte and data phase:
0: DDR Mode disabled
1: DDR Mode enabled
This field can be written only when BUSY = 0.

Bit 30 **DHHC**: DDR hold

Delay the data output by 1/4 of the QUADSPI output clock cycle in DDR mode:
0: Delay the data output using analog delay
1: Delay the data output by 1/4 of a QUADSPI output clock cycle.
This feature is only active in DDR mode.
This field can be written only when BUSY = 0.

Note: PRESCALER>0 is mandatory when DHHC=1.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **SIOO**: Send instruction only once mode

See *Section 20.3.12: Sending the instruction only once on page 586*. This bit has no effect when IMODE = 00.

0: Send instruction on every transaction

1: Send instruction only for the first command

This field can be written only when BUSY = 0.

Bits 27:26 **FMODE[1:0]**: Functional mode

This field defines the QUADSPI functional mode of operation.

00: Indirect write mode

01: Indirect read mode

10: Automatic polling mode

11: Memory-mapped mode

If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE value.

This field can be written only when BUSY = 0.

Bits 25:24 **DMODE[1:0]**: Data mode

This field defines the data phase’s mode of operation:

00: No data

01: Data on a single line

10: Data on two lines

11: Data on four lines

This field also determines the dummy phase mode of operation.

This field can be written only when BUSY = 0.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **DCYC[4:0]**: Number of dummy cycles

This field defines the duration of the dummy phase. In both SDR and DDR modes, it specifies a number of CLK cycles (0-31).

This field can be written only when BUSY = 0.

Bits 17:16 **ABSIZE[1:0]**: Alternate bytes size

This bit defines alternate bytes size:

00: 8-bit alternate byte

01: 16-bit alternate bytes

10: 24-bit alternate bytes

11: 32-bit alternate bytes

This field can be written only when BUSY = 0.

Bits 15:14 **ABMODE[1:0]**: Alternate bytes mode

This field defines the alternate-bytes phase mode of operation:

00: No alternate bytes

01: Alternate bytes on a single line

10: Alternate bytes on two lines

11: Alternate bytes on four lines

This field can be written only when BUSY = 0.

Bits 13:12 **ADSIZE[1:0]**: Address size

This bit defines address size:
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address
This field can be written only when BUSY = 0.

Bits 11:10 **ADMODE[1:0]**: Address mode

This field defines the address phase mode of operation:
00: No address
01: Address on a single line
10: Address on two lines
11: Address on four lines
This field can be written only when BUSY = 0.

Bits 9:8 **IMODE[1:0]**: Instruction mode

This field defines the instruction phase mode of operation:
00: No instruction
01: Instruction on a single line
10: Instruction on two lines
11: Instruction on four lines
This field can be written only when BUSY = 0.

Bits 7:0 **INSTRUCTION[7:0]**: Instruction

Instruction to be send to the external SPI device.
This field can be written only when BUSY = 0.

20.5.7 QUADSPI address register (QUADSPI_AR)

Address offset: 0x0018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[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
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address

Address to be send to the external Flash memory
Writes to this field are ignored when BUSY = 1 or when FMODE = 11 (memory-mapped mode).
In dual flash mode, ADDRESS[0] is automatically stuck to ‘0’ as the address should always be even

20.5.8 QUADSPI alternate bytes registers (QUADSPI_ABR)

Address offset: 0x001C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[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
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate Bytes

Optional data to be send to the external SPI device right after the address.

This field can be written only when BUSY = 0.

20.5.9 QUADSPI data register (QUADSPI_DR)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[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
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device.

In indirect write mode, data written to this register is stored on the FIFO before it is sent to the Flash memory during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect read mode, reading this register gives (via the FIFO) the data which was received from the Flash memory. If the FIFO does not have as many bytes as requested by the read operation and if BUSY=1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic polling mode, this register contains the last data read from the Flash memory (without masking).

Word, halfword, and byte accesses to this register are supported. In indirect write mode, a byte write adds 1 byte to the FIFO, a halfword write 2, and a word write 4. Similarly, in indirect read mode, a byte read removes 1 byte from the FIFO, a halfword read 2, and a word read 4. Accesses in indirect mode must be aligned to the bottom of this register: a byte read must read DATA[7:0] and a halfword read must read DATA[15:0].

20.5.10 QUADSPI polling status mask register (QUADSPI_PSMKR)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK[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
MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in polling mode.

For bit n:

0: Bit n of the data received in automatic polling mode is masked and its value is not considered in the matching logic

1: Bit n of the data received in automatic polling mode is unmasked and its value is considered in the matching logic

This field can be written only when BUSY = 0.

20.5.11 QUADSPI polling status match register (QUADSPI_PSMAR)

Address offset: 0x0028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MATCH[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
MATCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MATCH[31:0]**: Status match

Value to be compared with the masked status register to get a match.

This field can be written only when BUSY = 0.

20.5.12 QUADSPI polling interval register (QUADSPI_PIR)

Address offset: 0x002C

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
INTERVAL[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 **INTERVAL[15:0]**: Polling interval

Number of CLK cycles between to read during automatic polling phases.
This field can be written only when BUSY = 0.

20.5.13 QUADSPI low-power timeout register (QUADSPI_LPTR)

Address offset: 0x0030

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
TIMEOUT[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 **TIMEOUT[15:0]**: Timeout period

After each access in memory-mapped mode, the QUADSPI prefetches the subsequent bytes and holds these bytes in the FIFO. This field indicates how many CLK cycles the QUADSPI waits after the FIFO becomes full until it raises nCS, putting the Flash memory in a lower-consumption state.
This field can be written only when BUSY = 0.

20.5.14 QUADSPI register map

Table 157. QUADSPI 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	QUADSPI_CR	PRESCALER[7:0]								PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	Res.	FTHRES [3:0]				FSEL	DFM	Res.	SSHIFT	TCEN	DMAEN	ABORT	EN
	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
0x0004	QUADSPI_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSIZE[4:0]					Res.	Res.	Res.	Res.	Res.	CSHT			Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKMODE
	Reset value												0	0	0	0	0						0	0	0								0
0x0008	QUADSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[4:0]					Res.	Res.	BUS	TOF	SMF	FTF	TCF	TEF
	Reset value																				0	0	0	0	0			0	0	0	0	0	0
0x000C	QUADSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																												0	0		0	0
0x0010	QUADSPI_DLR	DL[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
0x0014	QUADSPI_CCR	DDRM	DHHC	Res.	SIOO	FMODE[1:0]		DMODE[1:0]		Res.	DCYC[4:0]					ABSIZE[1:0]		ABMODE[1:0]		ADSIZE[1:0]		ADMODE[1:0]		IMODE[1:0]		INSTRUCTION[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
0x0018	QUADSPI_AR	ADDRESS[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
0x001C	QUADSPI_ABR	ALTERNATE[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
0x0020	QUADSPI_DR	DATA[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
0x0024	QUADSPI_PSMKR	MASK[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
0x0028	QUADSPI_PSMAR	MATCH[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
0x002C	QUADSPI_PIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0030	QUADSPI_LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 81* for the register boundary addresses.

21 Analog-to-digital converters (ADC)

21.1 Introduction

This section describes the implementation of up to 5 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).
- ADC3 and ADC4 are tightly coupled and can operate in dual mode (ADC3 is master).
- ADC5 is controlled independently.

Each ADC consists of a 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 19 multiplexed channels. 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 ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows to improve analog performance while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

21.2 ADC main features

- High-performance features
  - Up to 5 ADCs, out of which four of them ((in pairs) can operate in dual mode:
    ADC1 is connected to 14 external channels + 4 internal channels
    ADC2 is connected to 16 external channels + 2 internal channels
    ADC3 is connected to 15 external channels + 3 internal channels
    ADC4 is connected to 16 external channels + 2 internal channels
    ADC5 is connected to 13 external channels + 5 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time is independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Manage single-ended or differential inputs
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Up to four injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - 4 dedicated data registers for the injected channels
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256
  - Programmable data shift up to 8-bit
- Data preconditioning
  - Gain compensation
  - Offset compensation
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency

application (auto-delayed mode)

- Number of external analog input channels per ADC
  - Up to 5 fast channels from GPIO pads
  - Up to 13 slow channels from GPIO pads
- In addition, there are several internal dedicated channels
  - The internal reference voltage (V_{REFINT}), connected to ADC1, 3, 4 and 5
  - The internal temperature sensor (V_{TS}), connected to ADC1 and 5
  - The V_{BAT} monitoring channel ($V_{BAT}/3$), connected to ADC1, 3 (for category 3 devices only) and 5
  - The OPAMP1 internal output connected to ADC1
  - The OPAMP2 and OPAMP3 internal outputs connected to ADC2
  - The OPAMP3 internal output connected to ADC3
  - The OPAMP6 internal output connected to ADC4 (for category 3 devices) or ADC3 (for category 4 devices)
  - The OPAMP4 and OPAMP5 internal outputs connected to ADC5
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC 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
- Dual ADC mode for ADC1, ADC2, ADC3 and ADC4
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
  - Watchdog can perform filtering to ignore out-of-range data
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

Figure 82 shows the block diagram of one ADC.

Refer to the OPAMP electrical characteristics section of the product datasheet for the ADC sampling time value to be applied when converting the OPAMP output voltage.

21.3 ADC implementation

Table 158. ADC features

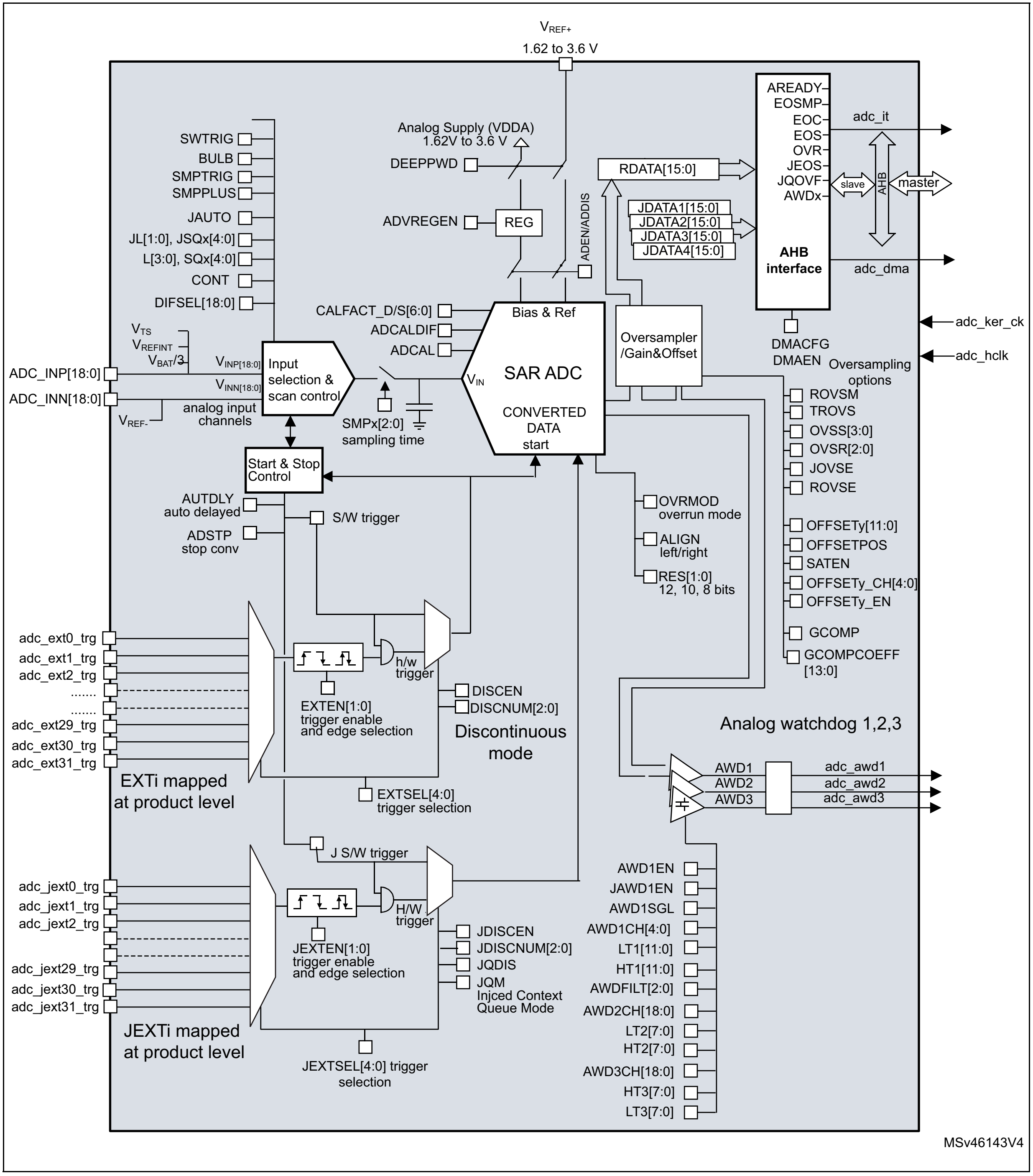
ADC modes/features	ADC1	ADC2	ADC3	ADC4	ADC5
Dual mode	X (coupled together)		X (coupled together)		-

21.4 ADC functional description

21.4.1 ADC block diagram

Figure 82 shows the ADC block diagram and *Table 160* gives the ADC pin description.

Figure 82. ADC block diagram



21.4.2 ADC pins and internal signals

Table 159. ADC internal input/output signals

Internal signal name	Signal type	Description
adc_ext_trg[31:0]	Inputs	Up to 32 external trigger inputs for the regular conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
adc_jext_trg[31:0]	Inputs	Up to 31 external trigger inputs for the injected conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
adc_awdx_out	Output	Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)
adc_ker_ck	Output	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request
V_{TS}	Input	Output voltage from internal temperature sensor
V_{REFINT}	Input	Output voltage from internal reference voltage
V_{BAT}	Input supply	External battery voltage supply

Table 160. ADC input/output pins

Pin name	Signal type	Comments
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC, 1.62 V ≤ V_{REF+} ≤ V_{DDA}
VDDA	Input, analog supply	Analog power supply equal V_{DDA}: 1.62 V ≤ V_{DDA} ≤ 3.6 V
VREF−	Input, analog reference negative	The lower/negative reference voltage for the ADC. V_{REF-} is internally connected to V_{SSA}
VSSA	Input, analog supply ground	Ground for analog power supply. On device package which do not have a dedicated V_{SSA} pin, V_{SSA} is internally connected to V_{SS}.
$V_{INP}i$	Positive analog input channels for each ADC	Connected either to ADCx_INP*i* external channels or to internal channels. This input is converted in single-ended mode
$V_{INN}i$	Negative analog input channels for each ADC	Connected either to V_{REF-} or to external channels: ADCx_INNi and ADCx_INP[i+1].

Table 160. ADC input/output pins (continued)

Pin name	Signal type	Comments
ADCx_INNi	Negative external analog input signals	Up to 19 analog input channels (x = ADC number = 1, 2, 3, 4 or 5). Refer to *Section 21.4.4: ADC1/2/3/4/5 connectivity* for details.
ADCx_INPi	Positive external analog input signals	Up to 19 analog input channels (x = ADC number = 1, 2, 3, 4 or 5). Refer to *Section 21.4.4: ADC1/2/3/4/5 connectivity* for details

21.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The input clock is the same for all ADCs and can be selected between two different clock sources (see *Figure 83: ADC clock scheme*):

1. The ADC clock can be a specific clock source, derived from the following clock sources:
   – The system clock
   – PLL “P” clock

   Refer to RCC Section for more information on how to generate ADC dedicated clock. To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be reset.
2. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).

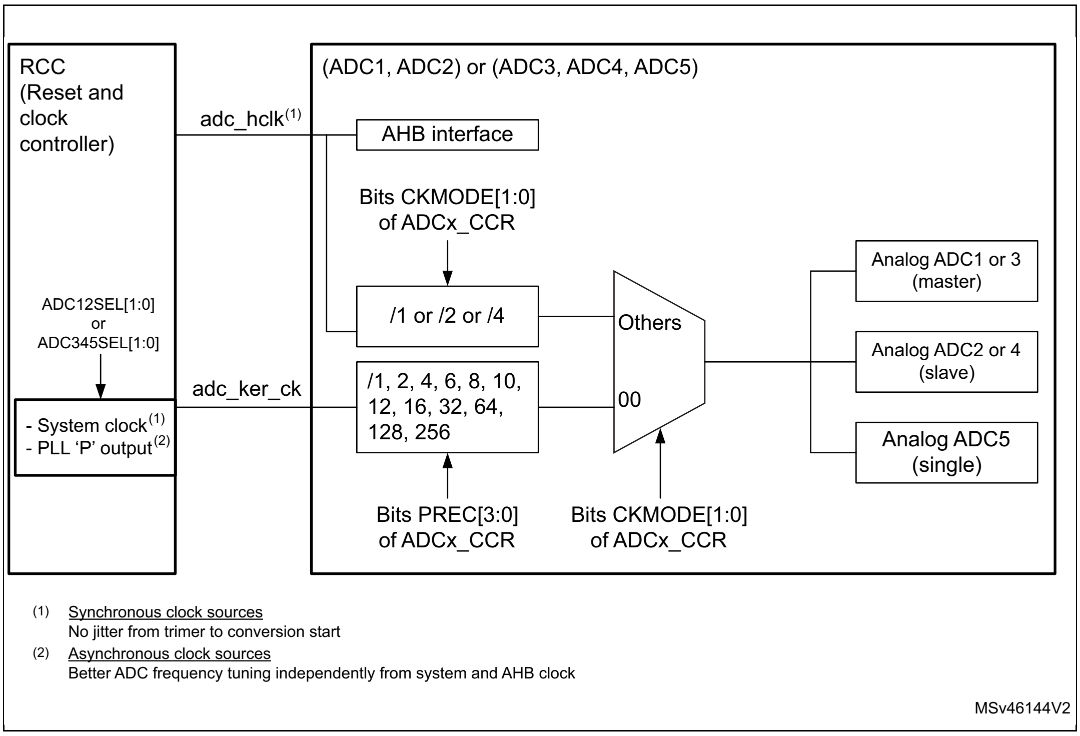
   To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be different from “00”.

Note: For option 2), a prescaling factor of 1 (CKMODE[1:0]=01) can be used only if the AHB prescaler is set to 1 (HPRE[3:0] = 0xxx in RCC_CFGR register).

Option 1) has the advantage of reaching the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256; using the prescaler configured with bits PRESC[3:0] in the ADCx_CCR register.

Option 2) 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 time is added by the resynchronizations between the two clock domains).

Figure 83. ADC clock scheme



(1) Synchronous clock sources
No jitter from trimer to conversion start

(2) Asynchronous clock sources
Better ADC frequency tuning independently from system and AHB clock

Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{adc\_hclk} \geq F_{ADC} / 4$ if the resolution of all channels are 12-bit or 10-bit
- $F_{adc\_hclk} \geq F_{ADC} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolution)
- $F_{adc\_hclk} \geq F_{ADC} / 2$ if there are some channels with resolutions equal to 6-bit

21.4.4 ADC1/2/3/4/5 connectivity

ADC1, ADC2, ADC3, ADC4 and ADC5 are tightly coupled and share some external channels as described in the below figures.

ADCy_INPx correspond to ADCy_INx pins defined in the product datasheet.

Figure 84. ADC1 connectivity

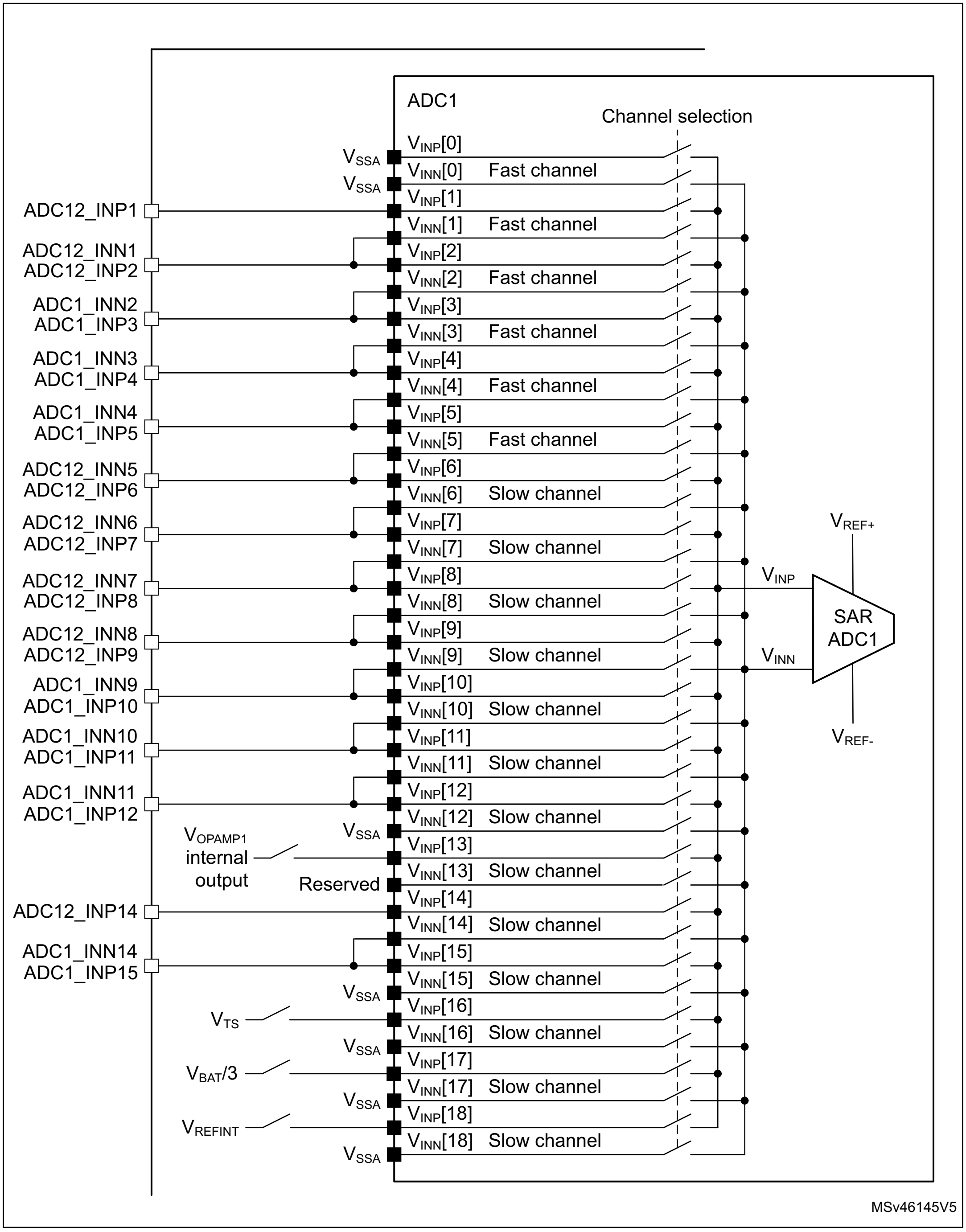


Figure 85. ADC2 connectivity

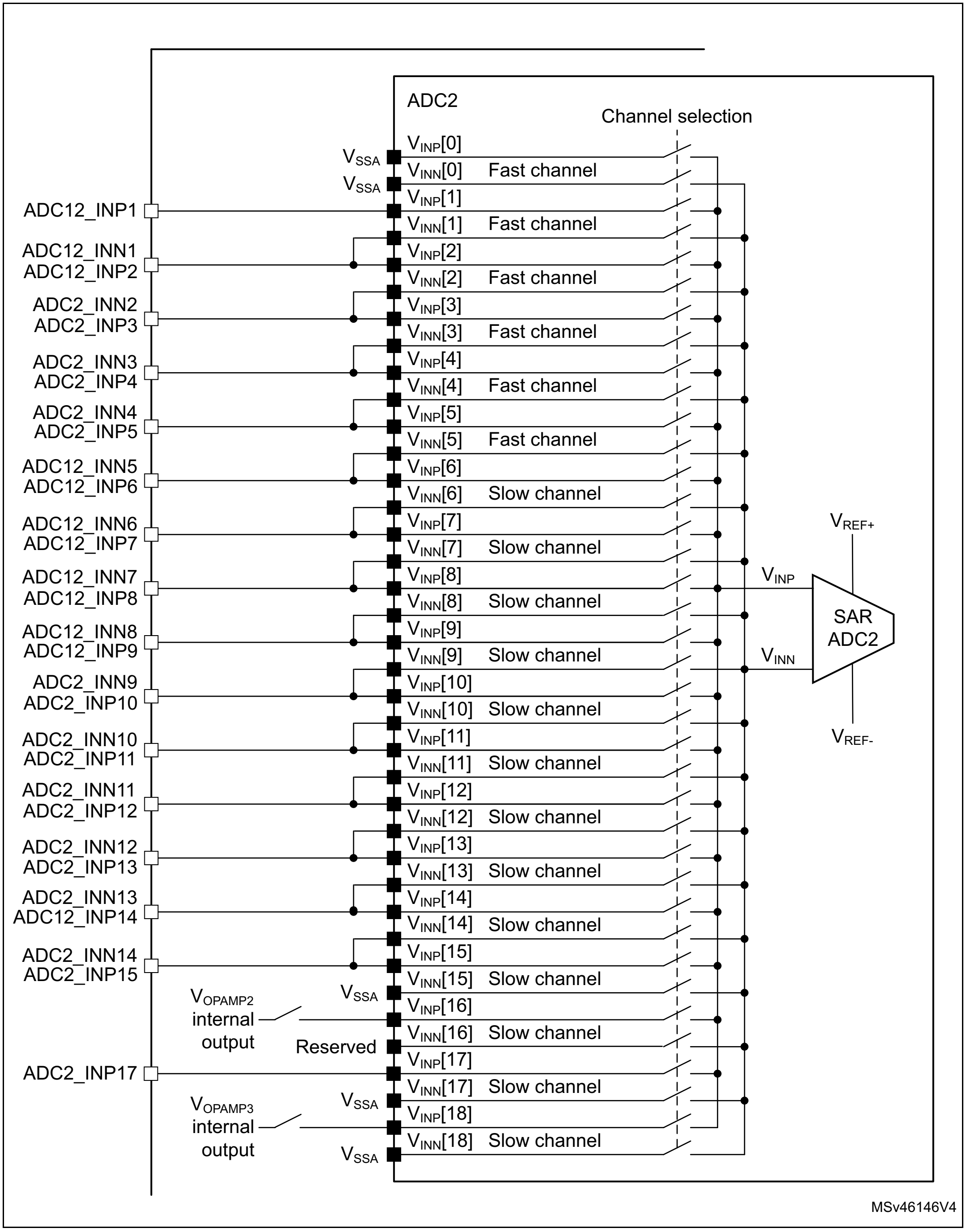
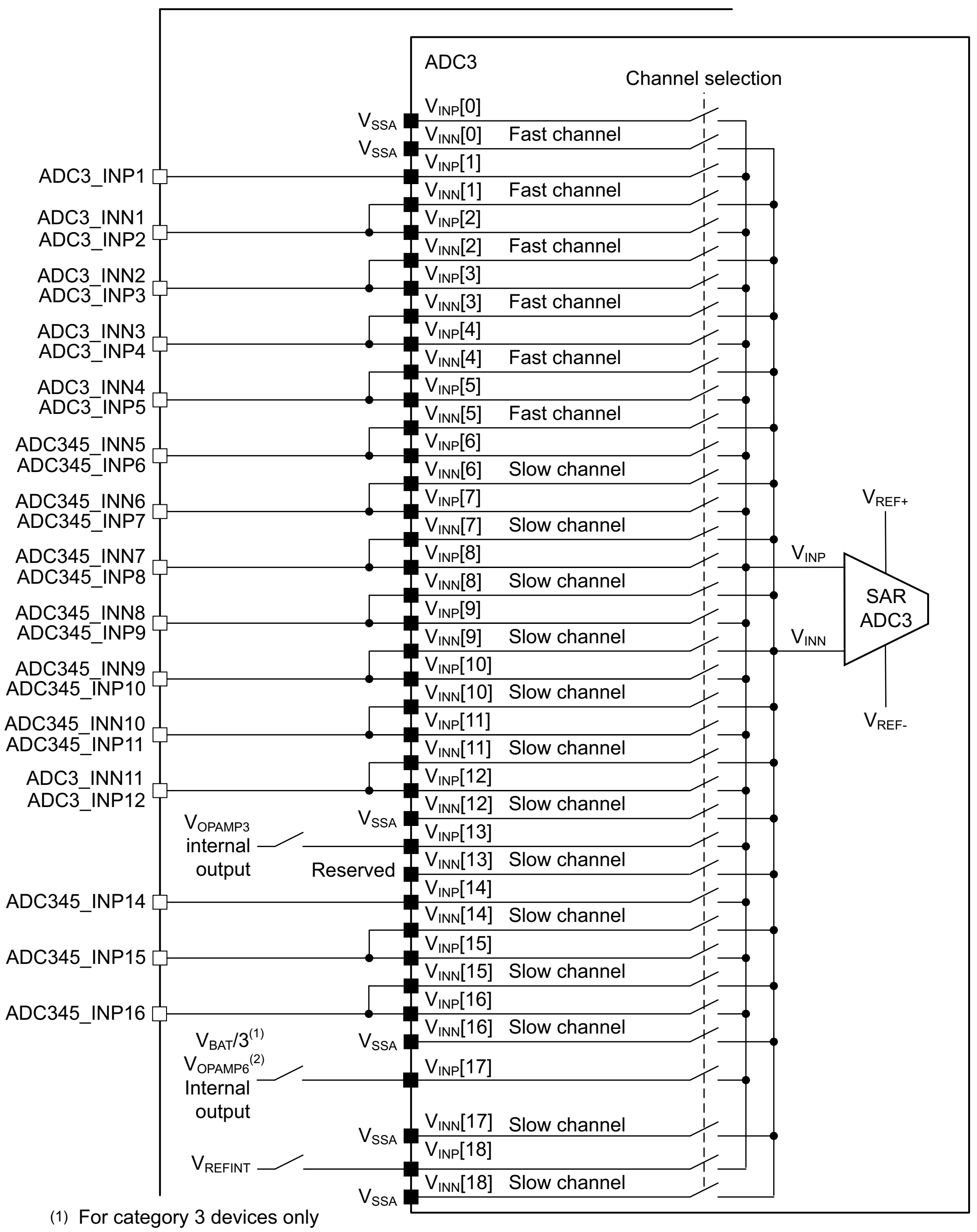


Figure 86. ADC3 connectivity



(1) For category 3 devices only

(2) For category 4 devices only

Note: in category 4 devices the ADC345_xxx pins correspond to ADC3_xxx pins.

Figure 87. ADC4 connectivity

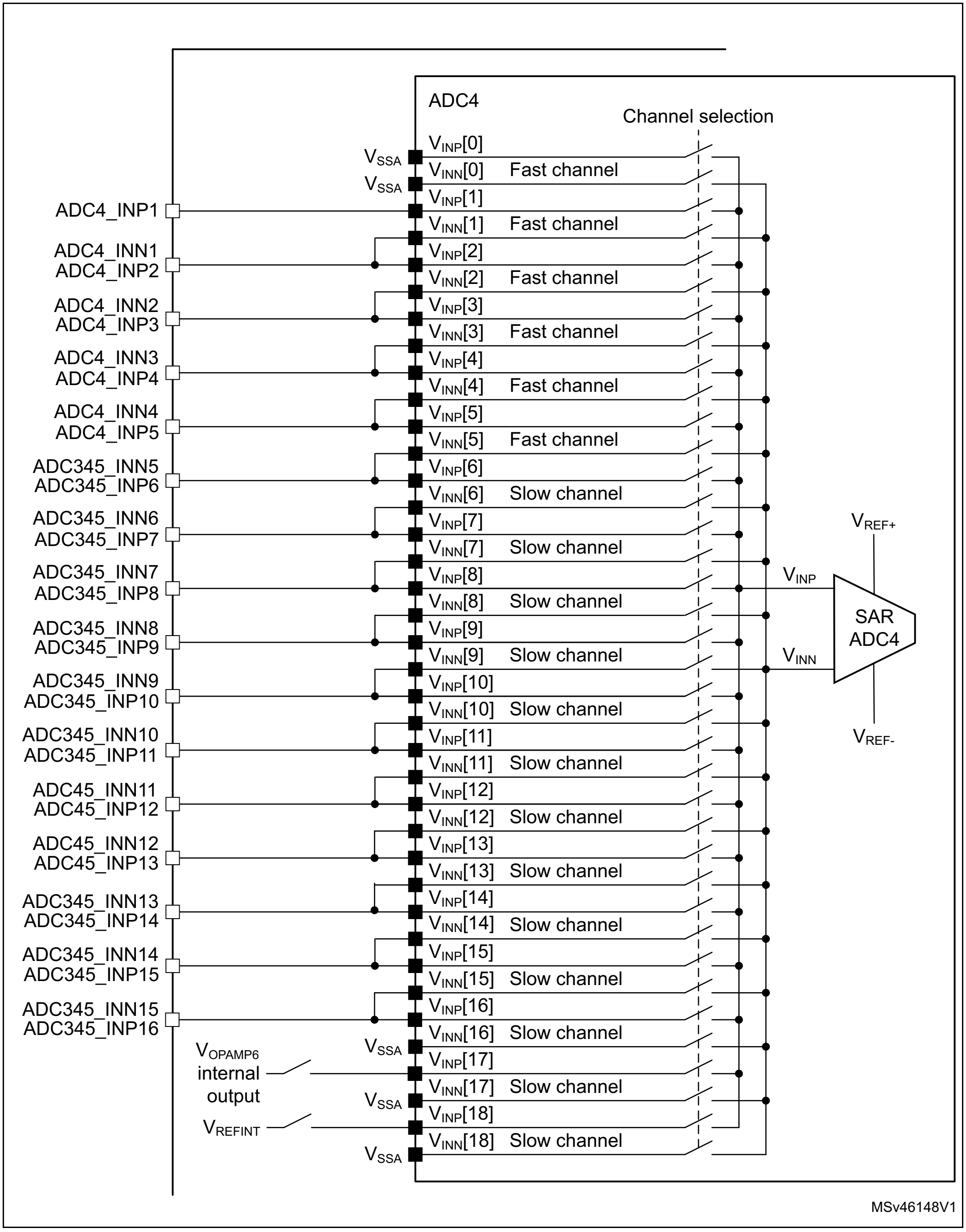
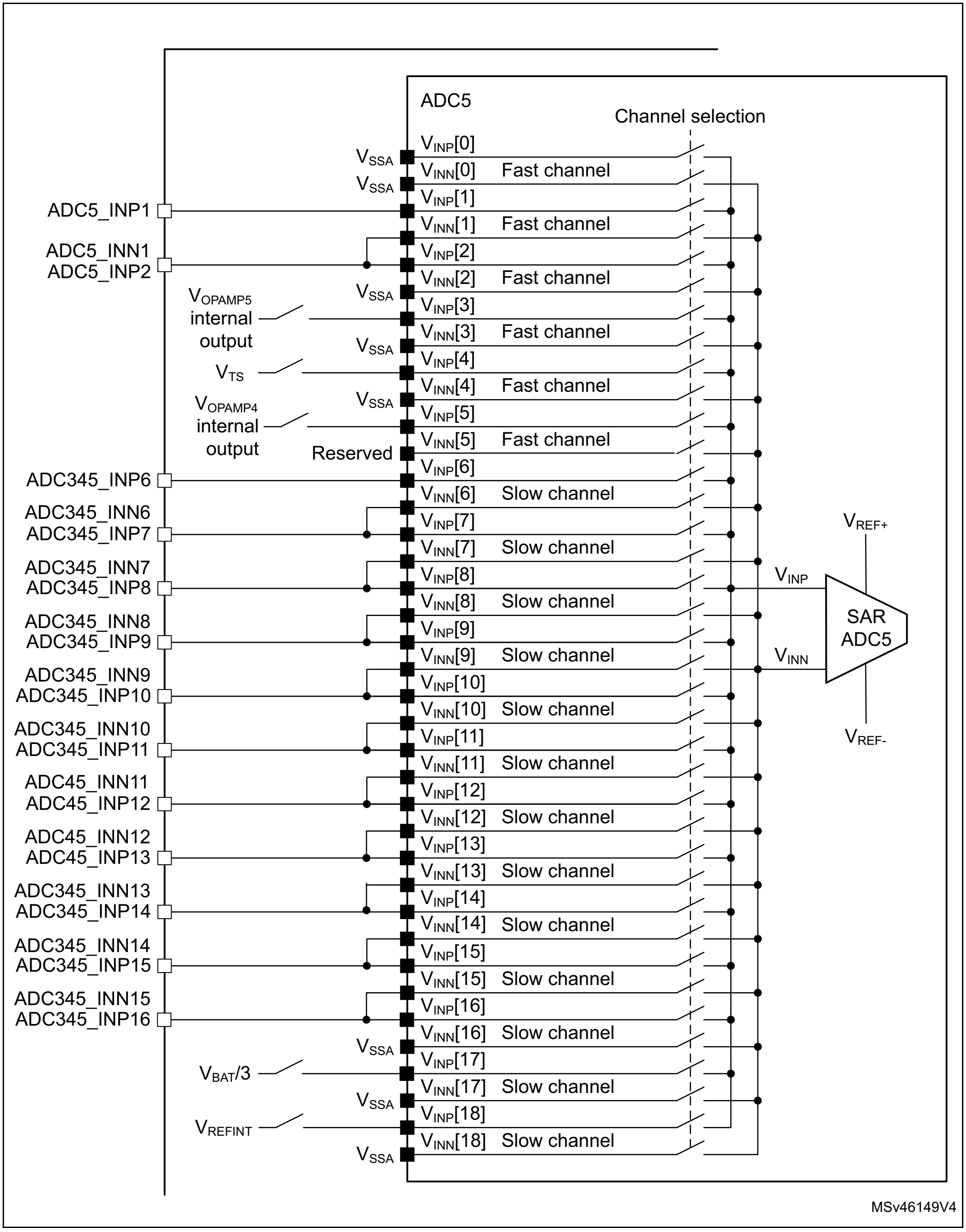


Figure 88. ADC5 connectivity



21.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

21.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in Deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC_CR register).

To start ADC operations, it is first needed to exit Deep-power-down mode by setting bit DEEPPWD=0.

When ADC operations are complete, the ADC can be disabled (ADEN=0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN=0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC Deep-power-down mode by setting bit DEEPPWD=1 into ADC_CR register. This is particularly interesting before entering Stop mode.

Note: Writing DEEPPWD=1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.

When the internal voltage regulator is disabled (ADVREGEN=0), the internal analog calibration is kept.

In ADC Deep-power-down mode (DEEPPWD=1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or re-apply the calibration factor which was previously saved (refer to *Section 21.4.8: Calibration (ADCAL, ADCALDIF, ADC_CALFACT)*).

21.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN=0). Note that the DIFSEL[i] bits corresponding to single-ended channels are always programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the ADCy_INPx external voltage equal to $V_{INP[i]}$ (positive input) and V_{REF-} (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the ADCy_INPx external voltage positive input equal to $V_{INP[i]}$, and the ADCy_INNx negative input equal to $V_{INN[i]}$.

The input voltage in differential mode ranges from V_{REF-} to V_{REF+}, which makes a full scale range of $2xV_{REF+}$. When $V_{INP[i]}$ equals V_{REF-}, $V_{INN[i]}$ equals $V_{REF+}$and the maximum negative input differential voltage (V_{REF-}) corresponds to 0x000 ADC output. When $V_{INP[i]}$ equals V_{REF+}, $V_{INN[i]}$ equals V_{REF-} and the maximum positive input differential voltage (V_{REF+}) corresponds to 0xFFF ADC output. When $V_{INP[i]}$ and $V_{INN[i]}$ are connected together, the zero input differential voltage corresponds to 0x800 ADC output.

The ADC sensitivity in differential mode is twice smaller than in single-ended mode.

When ADC is configured as differential mode, both inputs should be biased at (V_{REF+}) / 2 voltage. Refer to the device datasheet for the allowed common mode input voltage V_{CMIN}.

The input signals are supposed to be differential (common mode voltage should be fixed).

Internal channels (such as V_{TS} and V_{REFINT}) are used in single-ended mode only.

For a complete description of how the input channels are connected for each ADC, refer to *Section 21.4.4: ADC1/2/3/4/5 connectivity*.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage $V_{INN[i]}$ is connected to another channel. As a consequence, this channel is no longer usable in single-ended mode or in differential mode and must never be configured to be converted. Some channels are shared between ADC1/ADC2/ADC3/ADC4/ADC5: this can make the channel on the other ADC unusable. The only exception is when ADC master and the slave operate in interleaved mode.

21.4.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF=0 before launching a calibration which will be applied for single-ended input conversions.
- Write ADCALDIF=1 before launching a calibration which will be applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL=1. Calibration can only be initiated when the ADC is disabled (when ADEN=0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADC_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN=0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

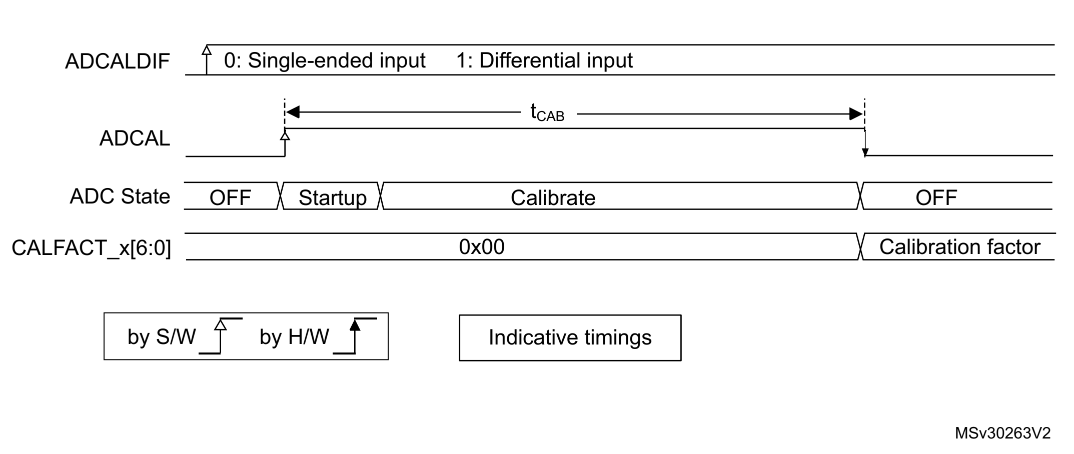
The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in Standby or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN=1 and ADSTART=0 and JADSTART=0). Then, at the next start of conversion, the calibration factor will automatically be injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when V_{REF+} voltage changed more than 10%.

Software procedure to calibrate the ADC

1. Ensure DEEPPWD=0, ADVREGEN=1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN=0.
3. Select the input mode for this calibration by setting ADCALDIF=0 (single-ended input) or ADCALDIF=1 (differential input).
4. Set ADCAL=1.
5. Wait until ADCAL=0.
6. The calibration factor can be read from ADC_CALFACT register.

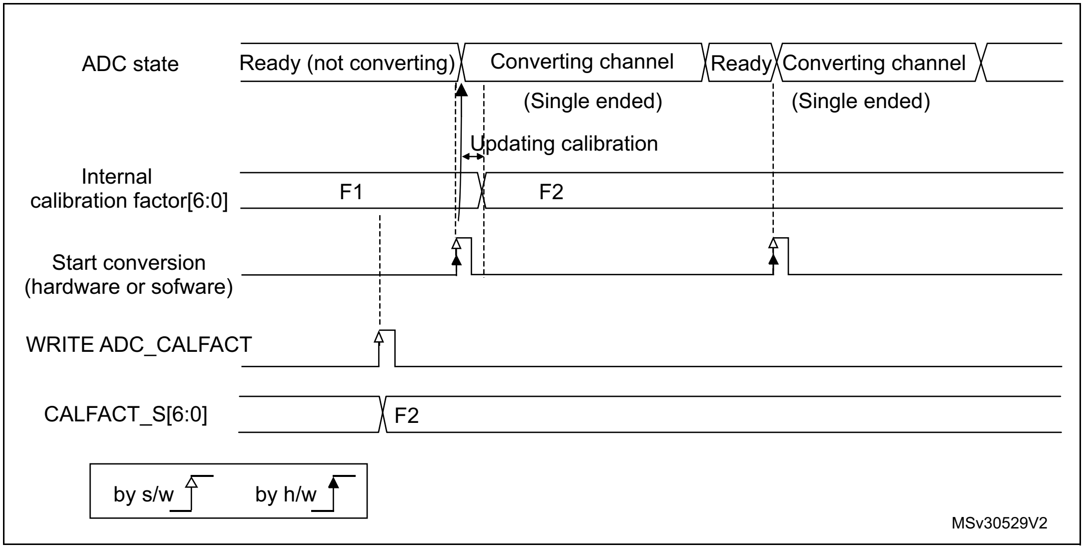
Figure 89. ADC calibration



Software procedure to re-inject a calibration factor into the ADC

1. Ensure ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor will be injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 90. Updating the ADC calibration factor

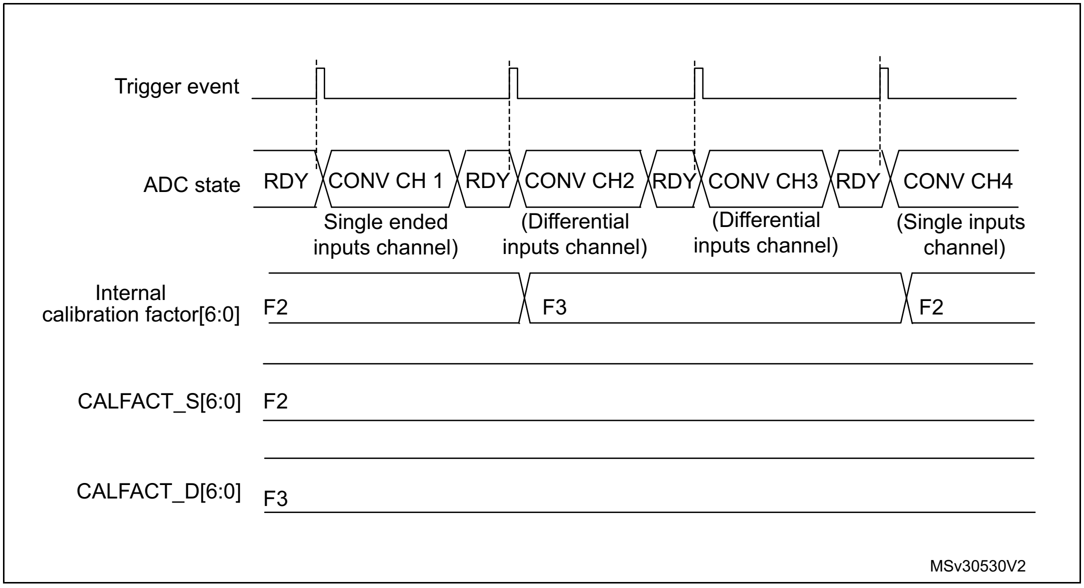


Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF=0 and one with ADCALDIF=1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF=0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF=1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration will automatically be injected into the analog ADC.

Figure 91. Mixing single-ended and differential channels



21.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in *Section 21.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD=0 and ADVREGEN=1, the ADC can be enabled and the ADC needs a stabilization time of t_{STAB} before it starts converting accurately, as shown in *Figure 92*. Two control bits enable or disable the ADC:

- ADEN=1 enables the ADC. The flag ADRDY will be set once the ADC is ready for operation.
- ADDIS=1 disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART=1 (refer to *Section 21.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART=1 or when an external injected trigger event occurs, if injected triggers are enabled.

Software procedure to enable the ADC

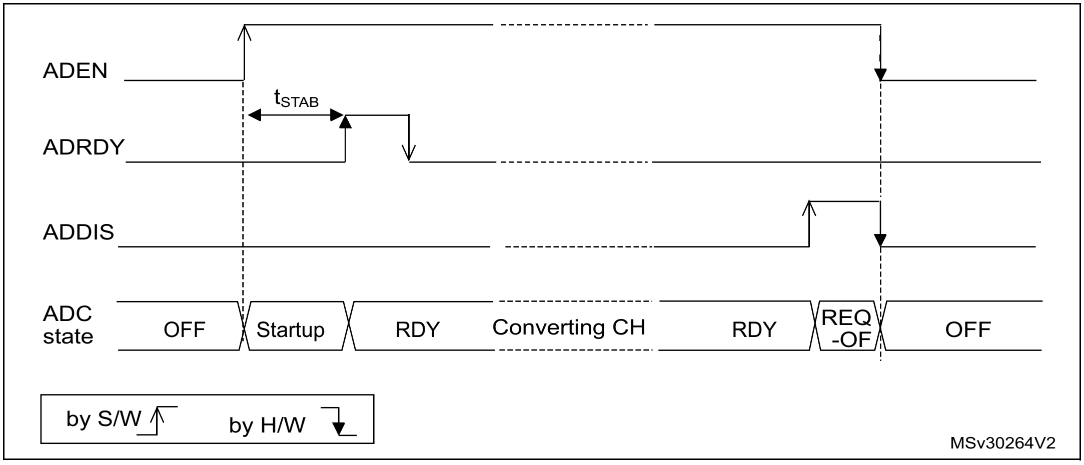
1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set ADEN=1.
3. Wait until ADRDY=1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE=1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

Caution: ADEN bit cannot be set when ADCAL is set and during four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).

Software procedure to disable the ADC

1. Check that both ADSTART=0 and JADSTART=0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP=1 and JADSTP=1 and then wait until ADSTP=0 and JADSTP=0.
2. Set ADDIS=1.
3. If required by the application, wait until ADEN=0, until the analog ADC is effectively disabled (ADDIS will automatically be reset once ADEN=0).

Figure 92. Enabling / disabling the ADC



21.4.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the DIFSEL[i] control bits in the ADC_DIFSEL register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).
- ADC_TRy registers can be modified when an analog-to-digital conversion is ongoing (refer to *Section 21.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*for details).

The software is allowed to write the ADSTP or JADSTP control bits of the ADC_CR register only if the ADC is enabled, possibly converting, and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN=1). Refer to *Section 21.6.16: ADC injected sequence register (ADC_JSQR)* for additional details.

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN=0 as well as all the bits of ADC_CR register).

21.4.11 Channel selection (SQRx, JSQRx)

There are up to 19 multiplexed channels per ADC:

- Up to 13 slow analog inputs coming from GPIO pads (ADCx_INP/INN[6:18])
  Depending on the products, not all of them are available on GPIO pads.
- The ADCs are connected to the following internal analog inputs:
  - The internal reference voltage (V_{REFINT}) is connected to ADC1_INP18, ADC3_INP18, ADC4_INP18 and ADC5_INP18.
  - The internal temperature sensor (V_{TS}) is connected to ADC1_INP16 and ADC5_INP4.
  - The V_{BAT} monitoring channel ($V_{BAT}/3$) is connected to ADC1_INP17, ADC3_INP17 and ADC5_INP17.

Note: To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, VBATSEL or VSENSESEL in the ADCx_CCR registers.

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_INP/INN3, ADCx_INP/INN8, ADCx_INP/INN2, ADCx_INN/INP2, ADCx_INP/INN0, ADCx_INP/INN2, ADCx_INP/INN2, ADCx_INP/INN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP=1 (refer to *Section 21.4.17: Stopping an ongoing conversion (ADSTP, JADSTP)*).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set to 1 (injected conversions ongoing) only when the context queue is enabled (JQDIS=0 in ADC_CFGR register). Refer to *Section 21.4.21: Queue of context for injected conversions*

21.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

T_{CONV} = Sampling time + 12.5 ADC clock cycles

Example:

With $F_{adc\_ker\_ck}$ = 30 MHz and a sampling time of 2.5 ADC clock cycles:

T_{CONV} = (2.5 + 12.5) ADC clock cycles = 15 ADC clock cycles = 500 ns

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

Bulb sampling mode

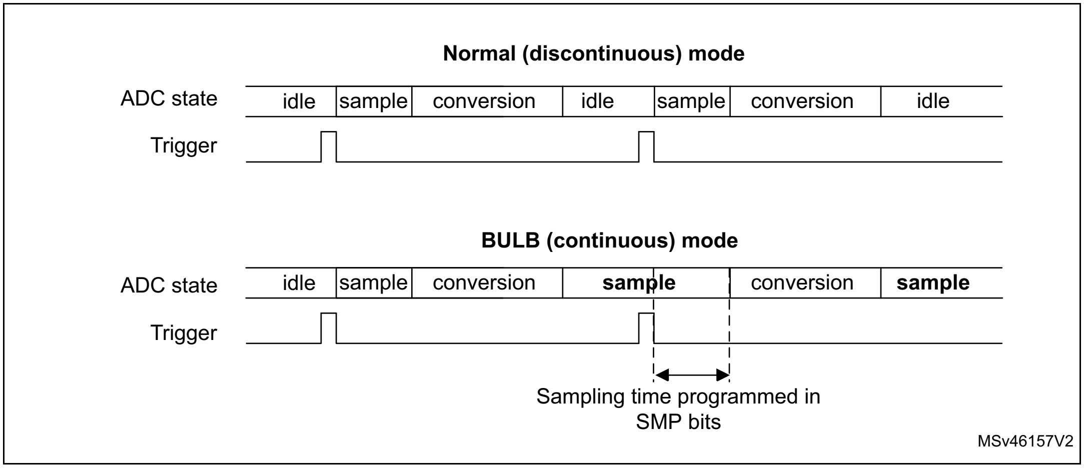
When the BULB bit is set in ADC register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPR1 register. The very first ADC conversion, after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The Bulb mode is effective starting from the second conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

The Bulb mode is neither compatible with the continuous conversion mode nor with the injected channel conversion.

When the BULB bit is set, it is not allowed to set SMPTRIG bit in ADC_CFGR2.

Figure 93. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed though SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected, each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion. The EXTEN[1:0] bits must be set to 01. Hardware triggers with not defined rising and falling edges (one pulse event) cannot be used in Bulb mode.

When a software trigger is selected, the software trigger is not the ADSTART bit in ADC_CR but the SWTRIG bit. SWTRIG bit has to be set to start the sampling period, and the SWTRIG bit has to be cleared to end the sampling period and start the conversion. EXTEN[1:0] bits must be set to 00.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is neither compatible with the continuous conversion mode, nor with the injected channel conversion.

When SMPTRIG bit is set, it is not allowed to set BULB bit.

I/O analog switches voltage booster

The I/O analog switches resistance increases when the V_{DDA} voltage is too low. This requires to have the sampling time adapted accordingly (cf datasheet for electrical characteristics). This resistance can be minimized at low V_{DDA} by enabling an internal voltage booster with BOOSTEN bit in the SYSCFG_CFGR1 register.

SMPPLUS control bit

When a sampling time of 2.5 ADC clock cycles is selected, the total conversion time becomes 15 cycles in 12-bit mode. If the dual interleaved mode is used (see *Section : Interleaved mode with independent injected*), the sampling interval cannot be equal to the value specified since an even number of cycles is required for the conversion. The SMPPLUS bit can be used to change the sampling time 2.5 ADC clock cycles into 3.5 ADC clock cycles. In this way, the total conversion time becomes 16 clock cycles, thus making possible to interleave every 8 cycles.

21.4.13 Single conversion mode (CONT=0)

In Single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

21.4.14 Continuous conversion mode (CONT=1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into 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 DISCEN=1 and CONT=1.

*Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section)***.**

21.4.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART=1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN[1:0] = 00 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN[1:0] is not equal to 00

Software starts ADC injected conversions by setting JADSTART=1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN[1:0] = 00 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN[1:0] is not equal to 00

Note: In auto-injection mode (JAUTO=1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART=0 and JADSTART=0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT=0, EXTSEL=0x0)
  - At any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT=x, EXTSEL=x)
  - After execution of the ADSTP procedure asserted by the software.

Note: In continuous mode (CONT=1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT=0 and EXTSEL≠0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:

- In single mode with software injected trigger (JEXTSEL = 0x0)
  - At any end of injected conversion sequence (JEOS assertion) or at any end of subgroup processing if JDISCEN = 1
- in all cases (JEXTSEL=x)
  - After execution of the JADSTP procedure asserted by the software.

Note: When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.

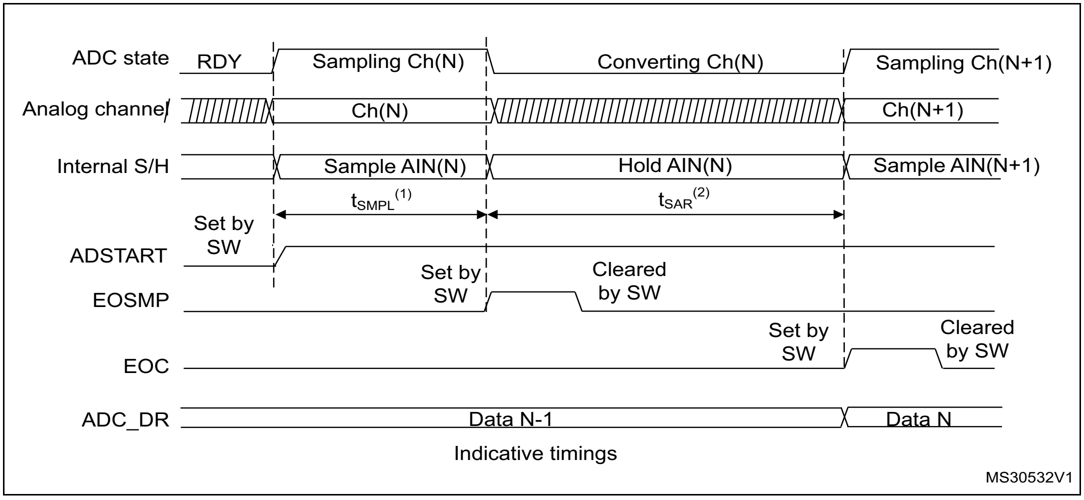
21.4.16 ADC timing

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:

$$T_{CONV} = T_{SMPL} + T_{SAR} = [2.5_{|min} + 12.5_{|12bit}] \times T_{ADC\_CLK}$$

$$T_{CONV} = T_{SMPL} + T_{SAR} = 83.33\text{ ns}_{|min} + 416.67\text{ ns}_{|12bit} = 500.0\text{ ns (for } F_{ADC\_CLK} = 30\text{ MHz)}$$

Figure 94. Analog to digital conversion time



1. T_{SMPL} depends on SMP[2:0].
2. T_{SAR} depends on RES[2:0].

21.4.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP=1 and injected conversions ongoing by setting JADSTP=1.

Stopping conversions will reset the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would restart a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 95. Stopping ongoing regular conversions

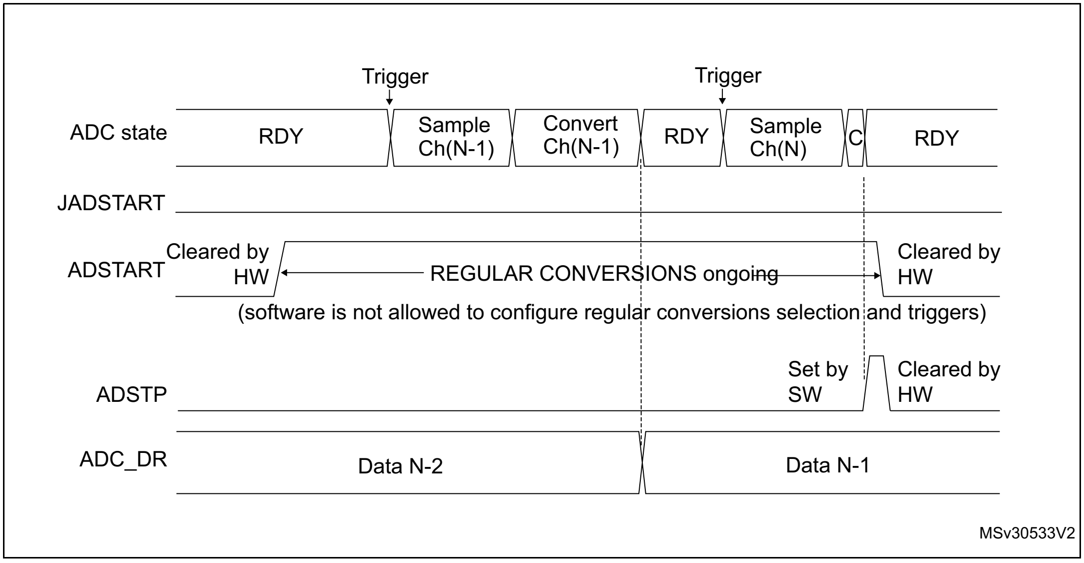
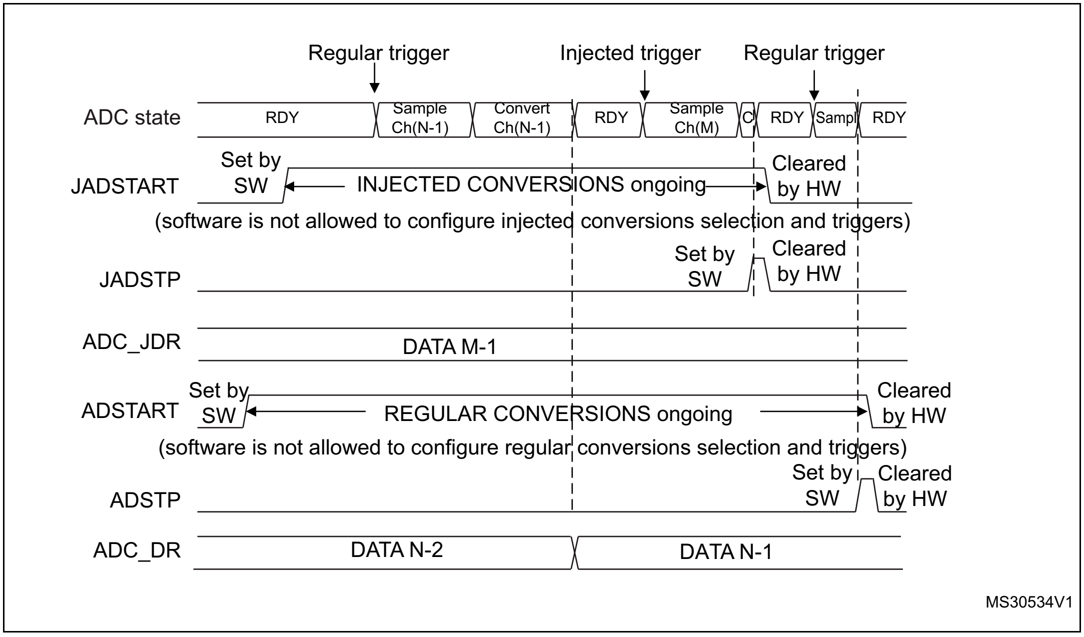


Figure 96. Stopping ongoing regular and injected conversions



21.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (e.g. timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS=0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART=1 and the injected trigger selection is effective once software has set bit JADSTART=1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART=0, any regular hardware triggers which occur are ignored.
- If bit JADSTART=0, any injected hardware triggers which occur are ignored.

Table 161 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 161. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Table 162. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	– If JQDIS=1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled – If JQDIS=0 (Queue enabled), Hardware and software trigger detection disabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS=0). Refer to Section 21.4.21: Queue of context for injected conversions.

The EXTSEL and JEXTSEL control bits select which out of 32 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly.
The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 21.4.21: Queue of context for injected conversions on page 636

Each ADC master shares the same input triggers with its ADC slave as described in *Figure 97*.

Figure 97. Triggers sharing between ADC master and ADC slave

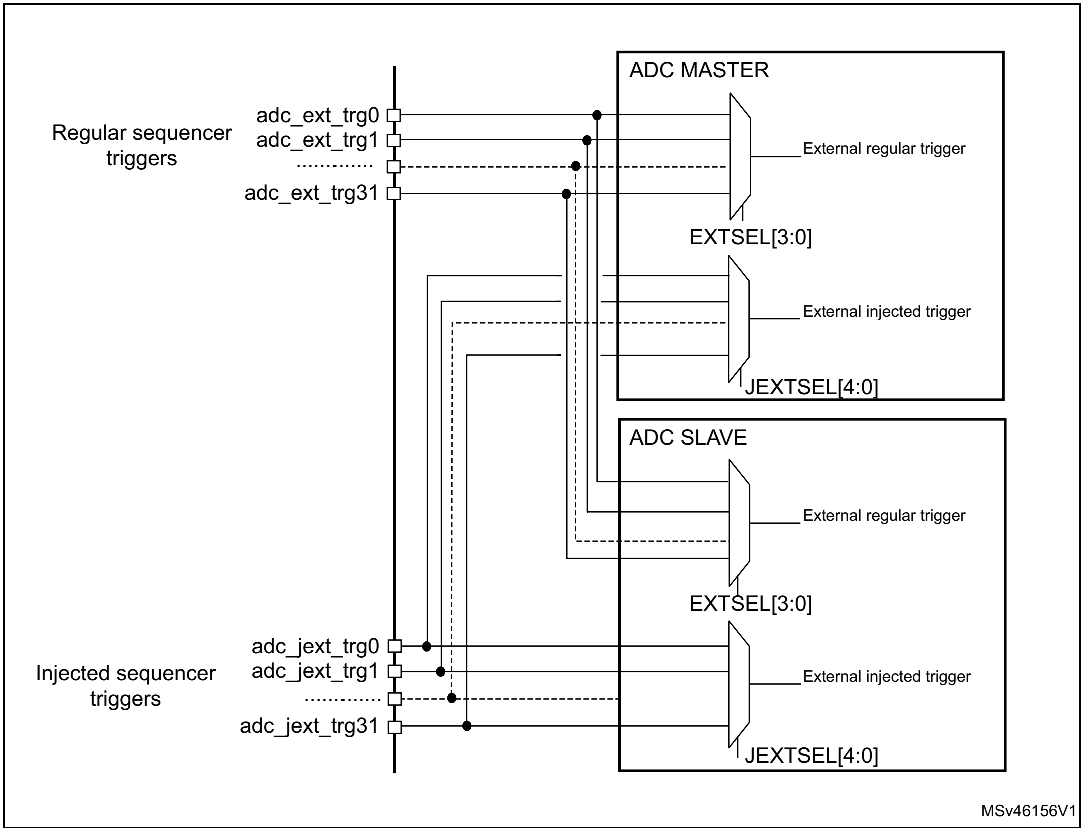


Table 163 to *Table 166* give all the possible external triggers of the three ADCs for regular and injected conversions.

Table 163. ADC1/2 - External triggers for regular channels

Name	Source	Type	EXTSEL[4:0]
adc_ext_trg	TIM1_CC1	Internal signal from on-chip timers	00000
adc_ext_trg1	TIM1_CC2	Internal signal from on-chip timers	00001
adc_ext_trg2	TIM1_CC3	Internal signal from on-chip timers	00010
adc_ext_trg3	TIM2_CC2	Internal signal from on-chip timers	00011
adc_ext_trg4	TIM3_TRGO	Internal signal from on-chip timers	00100
adc_ext_trg5	TIM4_CC4	Internal signal from on-chip timers	00101
adc_ext_trg6	EXTI line 11	External pin	00110
adc_ext_trg7	TIM8_TRGO	Internal signal from on-chip timers	00111
adc_ext_trg8	TIM8_TRGO2	Internal signal from on-chip timers	01000

Table 163. ADC1/2 - External triggers for regular channels (continued)

Name	Source	Type	EXTSEL[4:0]
adc_ext_trg9	TIM1_TRGO	Internal signal from on-chip timers	01001
adc_ext_trg10	TIM1_TRGO2	Internal signal from on-chip timers	01010
adc_ext_trg11	TIM2_TRGO	Internal signal from on-chip timers	01011
adc_ext_trg12	TIM4_TRGO	Internal signal from on-chip timers	01100
adc_ext_trg13	TIM6_TRGO	Internal signal from on-chip timers	01101
adc_ext_trg14	TIM15_TRGO	Internal signal from on-chip timers	01110
adc_ext_trg15	TIM3_CC4	Internal signal from on-chip timers	01111
adc_ext_trg16	TIM20_TRGO	Internal signal from on-chip timers	10000
adc_ext_trg17	TIM20_TRGO2	Internal signal from on-chip timers	10001
adc_ext_trg18	TIM20_CC1	Internal signal from on-chip timers	10010
adc_ext_trg19	TIM20_CC2	Internal signal from on-chip timers	10011
adc_ext_trg20	TIM20_CC3	Internal signal from on-chip timers	10100
adc_ext_trg21	hrtim_adc_trg1	Internal signal from on-chip timers	10101
adc_ext_trg22	hrtim_adc_trg3	Internal signal from on-chip timers	10110
adc_ext_trg23	hrtim_adc_trg5	Internal signal from on-chip timers	10111
adc_ext_trg24	hrtim_adc_trg6	Internal signal from on-chip timers	11000
adc_ext_trg25	hrtim_adc_trg7	Internal signal from on-chip timers	11001
adc_ext_trg26	hrtim_adc_trg8	Internal signal from on-chip timers	11010
adc_ext_trg27	hrtim_adc_trg9	Internal signal from on-chip timers	11011
adc_ext_trg28	hrtim_adc_trg10	Internal signal from on-chip timers	11100
adc_ext_trg29	LPTIMOUT	Internal signal from on-chip timers	11101
adc_ext_trg30	TIM7_TRGO	Internal signal from on-chip timers	11110
adc_ext_trg31	reserved	-	11111

Table 164. ADC1/2 - External trigger for injected channels

Name	Source	Type	JEXTSEL[4:0]
adc_jext_trg0	TIM1_TRGO	Internal signal from on-chip timers	00000
adc_jext_trg1	TIM1_CC4	Internal signal from on-chip timers	00001
adc_jext_trg2	TIM2_TRGO	Internal signal from on-chip timers	00010
adc_jext_trg3	TIM2_CC1	Internal signal from on-chip timers	00011
adc_jext_trg4	TIM3_CC4	Internal signal from on-chip timers	00100
adc_jext_trg5	TIM4_TRGO	Internal signal from on-chip timers	00101
adc_jext_trg6	EXTI line 15	External pin	00110
adc_jext_trg7	TIM8_CC4	Internal signal from on-chip timers	00111
adc_jext_trg8	TIM1_TRGO2	Internal signal from on-chip timers	01000

Table 164. ADC1/2 - External trigger for injected channels (continued)

Name	Source	Type	JEXTSEL[4:0]
adc_jext_trg9	TIM8_TRGO	Internal signal from on-chip timers	01001
adc_jext_trg10	TIM8_TRGO2	Internal signal from on-chip timers	01010
adc_jext_trg11	TIM3_CC3	Internal signal from on-chip timers	01011
adc_jext_trg12	TIM3_TRGO	Internal signal from on-chip timers	01100
adc_jext_trg13	TIM3_CC1	Internal signal from on-chip timers	01101
adc_jext_trg14	TIM6_TRGO	Internal signal from on-chip timers	01110
adc_jext_trg15	TIM15_TRGO	Internal signal from on-chip timers	01111
adc_jext_trg16	TIM20_TRGO	Internal signal from on-chip timers	10000
adc_jext_trg17	TIM20_TRGO2	Internal signal from on-chip timers	10001
adc_jext_trg18	TIM20_CC4	Internal signal from on-chip timers	10010
adc_jext_trg19	hrtim_adc_trg2	Internal signal from on-chip timers	10011
adc_jext_trg20	hrtim_adc_trg4	Internal signal from on-chip timers	10100
adc_jext_trg21	hrtim_adc_trg5	Internal signal from on-chip timers	10101
adc_jext_trg22	hrtim_adc_trg6	Internal signal from on-chip timers	10110
adc_jext_trg23	hrtim_adc_trg7	Internal signal from on-chip timers	10111
adc_jext_trg24	hrtim_adc_trg8	Internal signal from on-chip timers	11000
adc_jext_trg25	hrtim_adc_trg9	Internal signal from on-chip timers	11001
adc_jext_trg26	hrtim_adc_trg10	Internal signal from on-chip timers	11010
adc_jext_trg27	TIM16_CC1	Internal signal from on-chip timers	11011
adc_jext_trg28	reserved	-	11100
adc_jext_trg29	LPTIMOUT	Internal signal from on-chip timers	11101
adc_jext_trg30	TIM7_TRGO	Internal signal from on-chip timers	11110
adc_jext_trg31	reserved	-	11111

Table 165. ADC3/4/5 - External triggers for regular channels

Name	Source	Type	EXTSEL[4:0]
adc_ext_trg0	TIM3_CC1	Internal signal from on-chip timers	00000
adc_ext_trg1	TIM2_CC3	Internal signal from on-chip timers	00001
adc_ext_trg2	TIM1_CC3	Internal signal from on-chip timers	00010
adc_ext_trg3	TIM8_CC1	Internal signal from on-chip timers	00011
adc_ext_trg4	TIM3_TRGO	Internal signal from on-chip timers	00100
adc_ext_trg5	EXTI line 2	External pin	00101
adc_ext_trg6	TIM4_CC1	Internal signal from on-chip timers	00110
adc_ext_trg7	TIM8_TRGO	Internal signal from on-chip timers	00111

Table 165. ADC3/4/5 - External triggers for regular channels (continued)

Name	Source	Type	EXTSEL[4:0]
adc_ext_trg8	TIM8_TRGO2	Internal signal from on-chip timers	01000
adc_ext_trg9	TIM1_TRGO	Internal signal from on-chip timers	01001
adc_ext_trg10	TIM1_TRGO2	Internal signal from on-chip timers	01010
adc_ext_trg11	TIM2_TRGO	Internal signal from on-chip timers	01011
adc_ext_trg12	TIM4_TRGO	Internal signal from on-chip timers	01100
adc_ext_trg13	TIM6_TRGO	Internal signal from on-chip timers	01101
adc_ext_trg14	TIM15_TRGO	Internal signal from on-chip timers	01110
adc_ext_trg15	TIM2_CC1	Internal signal from on-chip timers	01111
adc_ext_trg16	TIM20_TRGO	Internal signal from on-chip timers	10000
adc_ext_trg17	TIM20_TRGO2	Internal signal from on-chip timers	10001
adc_ext_trg18	TIM20_CC1	Internal signal from on-chip timers	10010
adc_ext_trg19	hrtim_adc_trg2	Internal signal from on-chip timers	10011
adc_ext_trg20	hrtim_adc_trg4	Internal signal from on-chip timers	10100
adc_ext_trg21	hrtim_adc_trg1	Internal signal from on-chip timers	10101
adc_ext_trg22	hrtim_adc_trg3	Internal signal from on-chip timers	10110
adc_ext_trg23	hrtim_adc_trg5	Internal signal from on-chip timers	10111
adc_ext_trg24	hrtim_adc_trg6	Internal signal from on-chip timers	11000
adc_ext_trg25	hrtim_adc_trg7	Internal signal from on-chip timers	11001
adc_ext_trg26	hrtim_adc_trg8	Internal signal from on-chip timers	11010
adc_ext_trg27	hrtim_adc_trg9	Internal signal from on-chip timers	11011
adc_ext_trg28	hrtim_adc_trg10	Internal signal from on-chip timers	11100
adc_ext_trg29	LPTIMOUT	Internal signal from on-chip timers	11101
adc_ext_trg30	TIM7_TRGO	Internal signal from on-chip timers	11110
adc_ext_trg31	reserved	-	11111

Table 166. ADC3/4/5 - External triggers for injected channels

Name	Source	Type	JEXTSEL[4:0]
adc_jext_trg0	TIM1_TRGO	Internal signal from on-chip timers	00000
adc_jext_trg1	TIM1_CC4	Internal signal from on-chip timers	00001
adc_jext_trg2	TIM2_TRGO	Internal signal from on-chip timers	00010
adc_jext_trg3	TIM8_CC2	Internal signal from on-chip timers	00011
adc_jext_trg4	TIM4_CC3	Internal signal from on-chip timers	00100
adc_jext_trg5	TIM4_TRGO	Internal signal from on-chip timers	00101
adc_jext_trg6	TIM4_CC4	Internal signal from on-chip timers	00110
adc_jext_trg7	TIM8_CC4	Internal signal from on-chip timers	00111

Table 166. ADC3/4/5 - External triggers for injected channels (continued)

Name	Source	Type	JEXTSEL[4:0]
adc_jext_trg8	TIM1_TRGO2	Internal signal from on-chip timers	01000
adc_jext_trg9	TIM8_TRGO	Internal signal from on-chip timers	01001
adc_jext_trg10	TIM8_TRGO2	Internal signal from on-chip timers	01010
adc_jext_trg11	TIM1_CC3	Internal signal from on-chip timers	01011
adc_jext_trg12	TIM3_TRGO	Internal signal from on-chip timers	01100
adc_jext_trg13	EXTI line 3	External pin	01101
adc_jext_trg14	TIM6_TRGO	Internal signal from on-chip timers	01110
adc_jext_trg15	TIM15_TRGO	Internal signal from on-chip timers	01111
adc_jext_trg16	TIM20_TRGO	Internal signal from on-chip timers	10000
adc_jext_trg17	TIM20_TRGO2	Internal signal from on-chip timers	10001
adc_jext_trg18	TIM20_CC2	Internal signal from on-chip timers	10010
adc_jext_trg19	hrtim_adc_trg2	Internal signal from on-chip timers	10011
adc_jext_trg20	hrtim_adc_trg4	Internal signal from on-chip timers	10100
adc_jext_trg21	hrtim_adc_trg5	Internal signal from on-chip timers	10101
adc_jext_trg22	hrtim_adc_trg6	Internal signal from on-chip timers	10110
adc_jext_trg23	hrtim_adc_trg7	Internal signal from on-chip timers	10111
adc_jext_trg24	hrtim_adc_trg8	Internal signal from on-chip timers	11000
adc_jext_trg25	hrtim_adc_trg9	Internal signal from on-chip timers	11001
adc_jext_trg26	hrtim_adc_trg10	Internal signal from on-chip timers	11010
adc_jext_trg27	hrtim_adc_trg1	Internal signal from on-chip timers	11011
adc_jext_trg28	hrtim_adc_trg3	Internal signal from on-chip timers	11100
adc_jext_trg29	LPTIMOUT	Internal signal from on-chip timers	11101
adc_jext_trg30	TIM7_TRGO	Internal signal from on-chip timers	11110
adc_jext_trg31	reserved	-	11111

21.4.19 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is

reset and the injected channel sequence switches are launched (all the injected channels are converted once).

3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 98* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 2.5 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

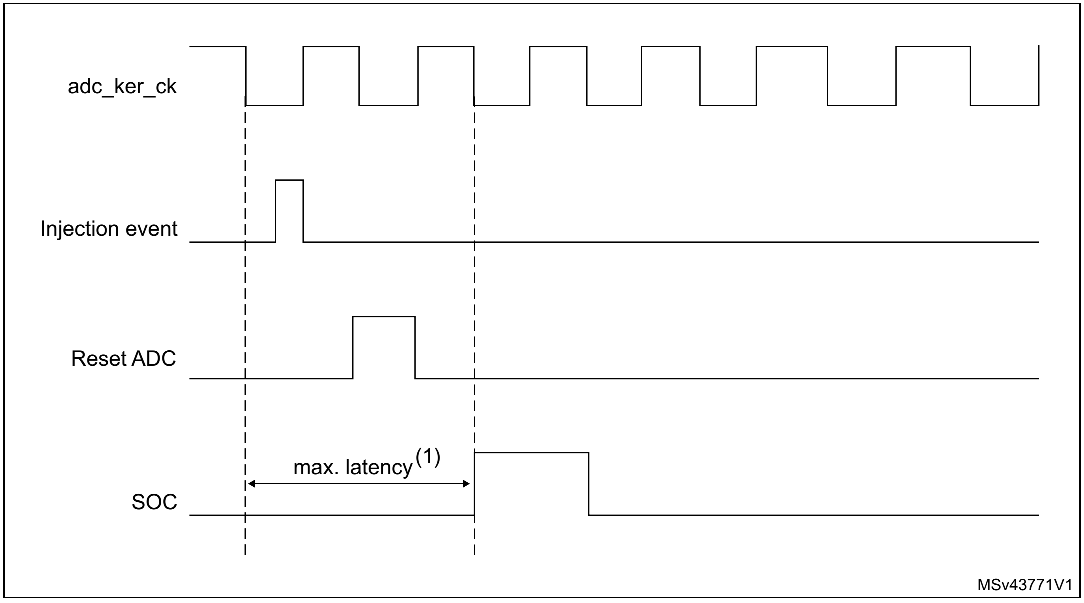
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence will be stopped upon DMA Transfer Complete event.

Figure 98. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

21.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (subgroup) of n conversions ($n \leq 8$) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN=1, n=3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN=0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events will relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN=1, CONT=1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where ‘n’ is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN=1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

21.4.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)

All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM=0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence will be served according to the last active context.
  - If JQM=1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP=1 or when disabling the ADC by setting ADDIS=1:
  - If JQM=0, the Queue is maintained with the last active context.
  - If JQM=1, the Queue becomes empty and triggers are ignored.

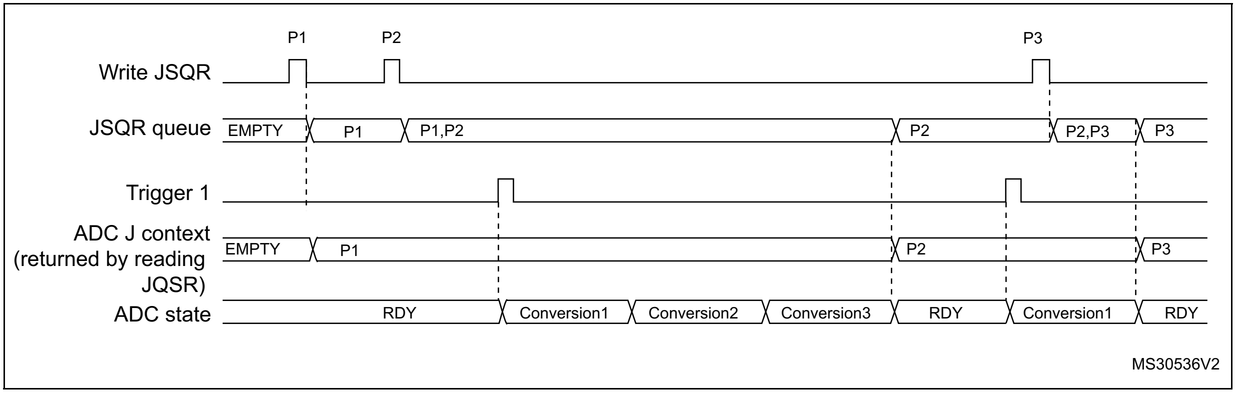
Note: When configured in discontinuous mode (bit JDISCEN=1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

- *1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out*
- *2nd trigger, disc. Sequence 1: 2nd conversion.*
- *3rd trigger, discontinuous. Sequence 1: 3rd conversion.*
- *4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.*
- *5th trigger, discontinuous. Sequence 2: 2nd conversion.*
- *6th trigger, discontinuous. Sequence 2: 3rd conversion.*

Behavior when changing the trigger or sequence context

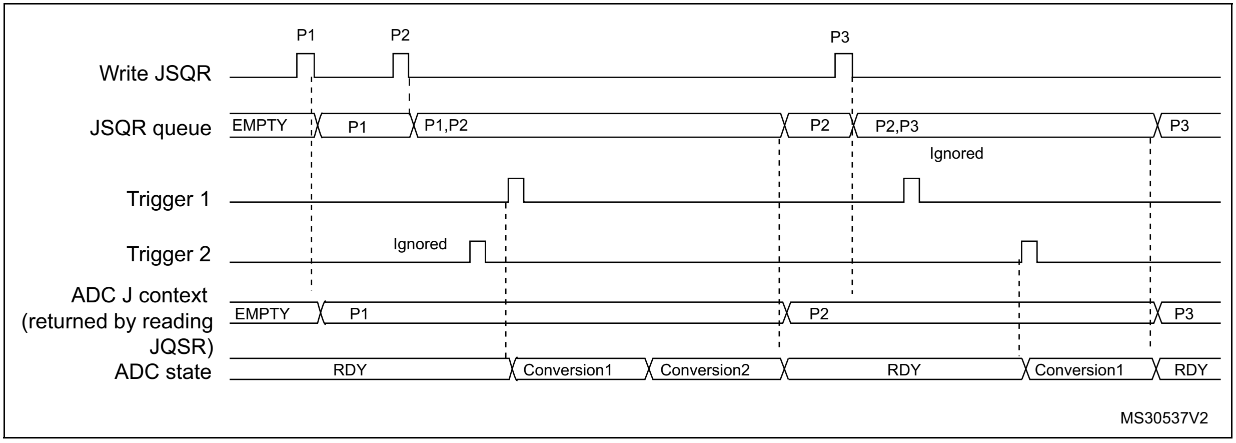
The *Figure 99* and *Figure 100* show the behavior of the context Queue when changing the sequence or the triggers.

Figure 99. Example of JSQR queue of context (sequence change)



1. Parameters:
   P1: sequence of 3 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 4 conversions, hardware trigger 1

Figure 100. Example of JSQR queue of context (trigger change)

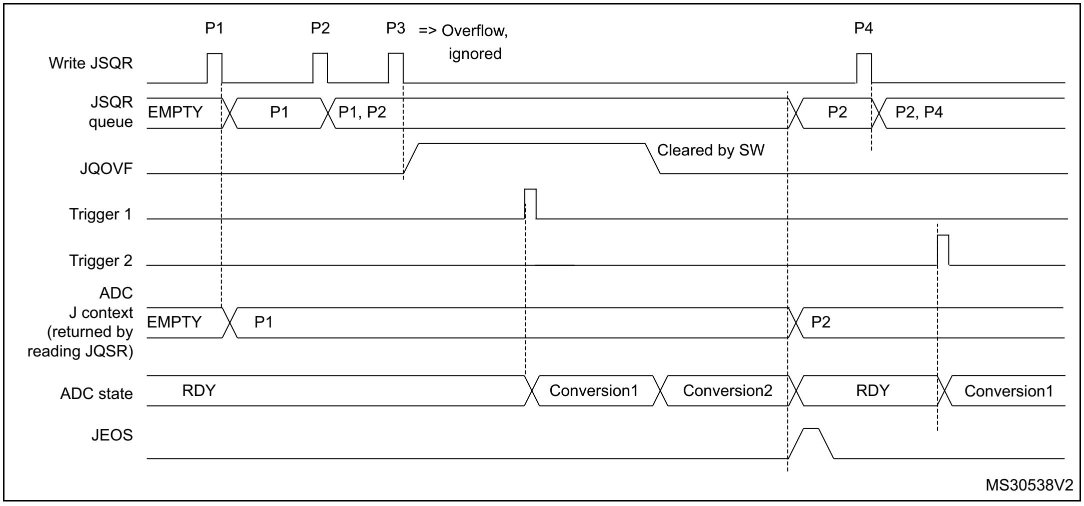


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

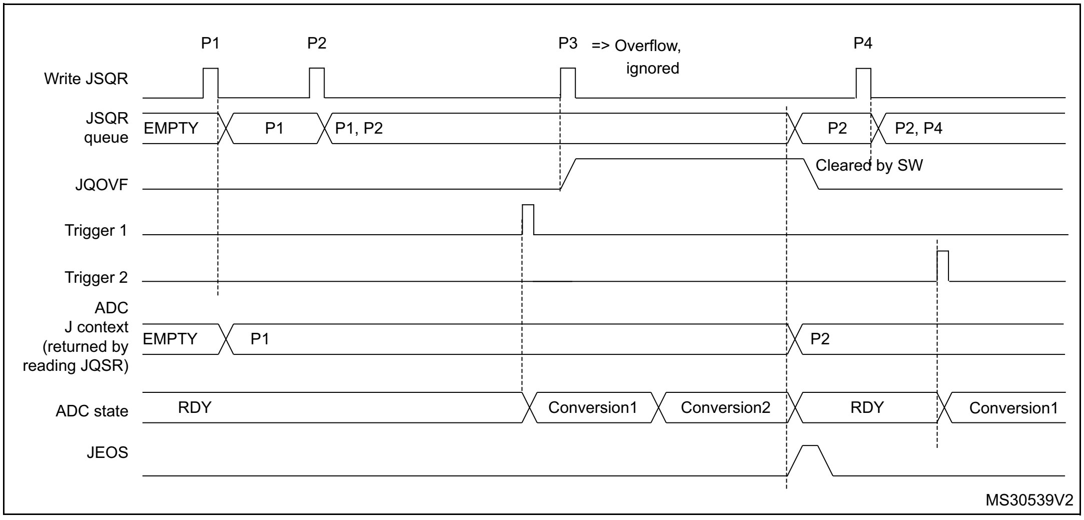
The *Figure 101* and *Figure 102* show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 101. Example of JSQR queue of context with overflow before conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

Figure 102. Example of JSQR queue of context with overflow during conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

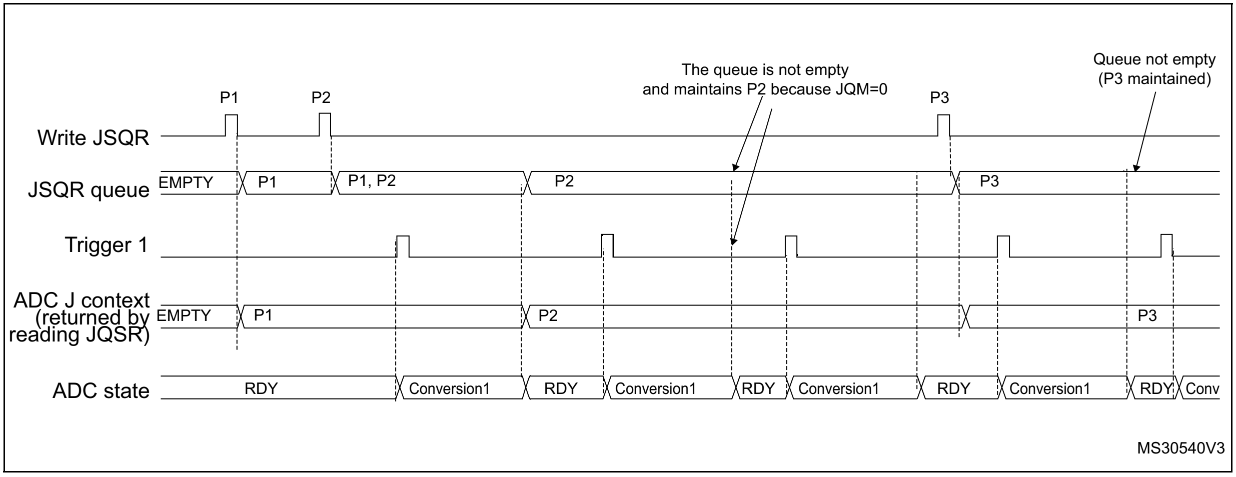
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 103 and *Figure 104* show the behavior of the context Queue when the Queue becomes empty in both cases JQM=0 or 1.

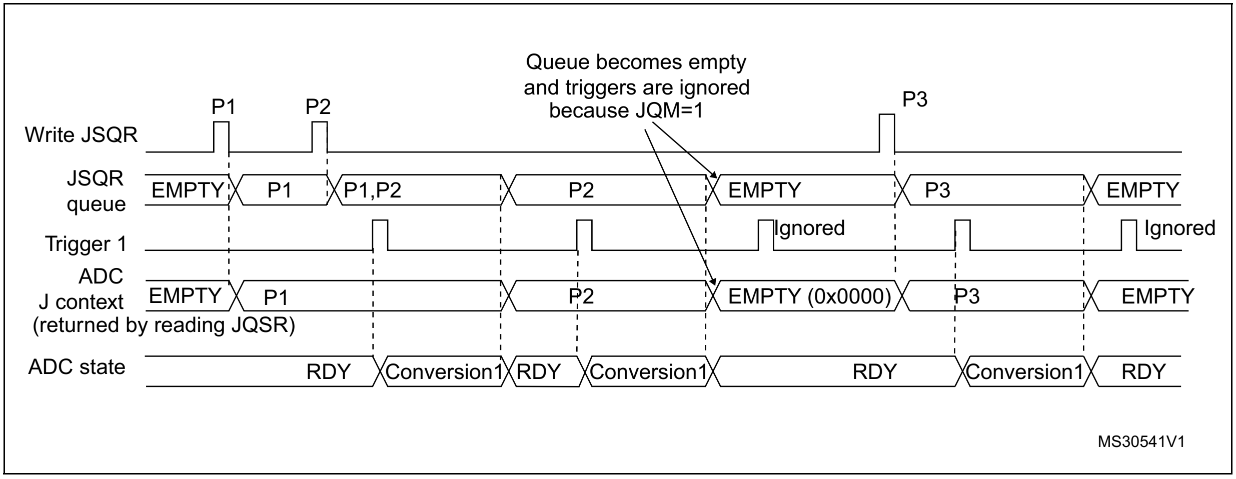
Figure 103. Example of JSQR queue of context with empty queue (case JQM=0)



1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 104. Example of JSQR queue of context with empty queue (case JQM=1)

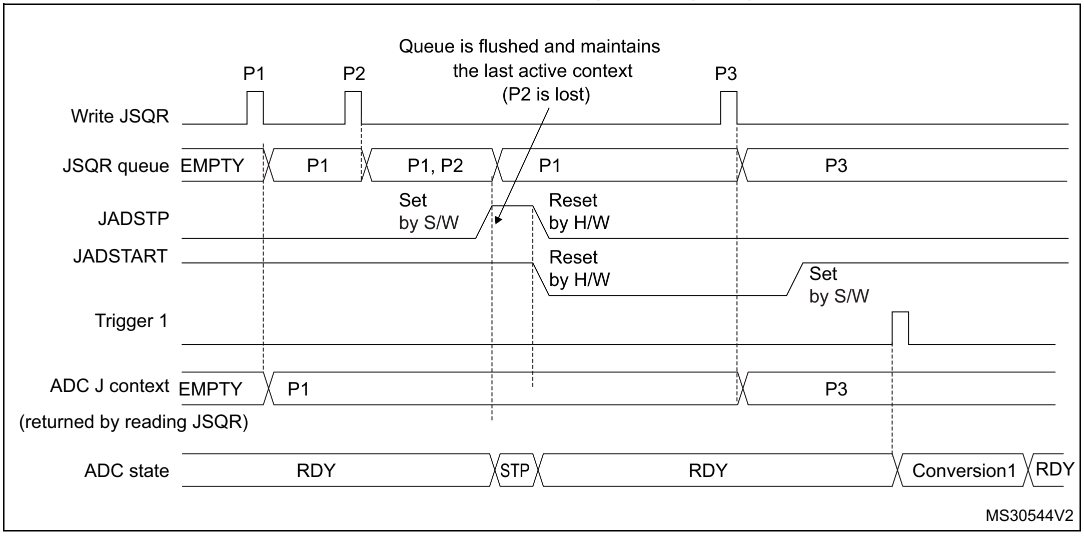


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

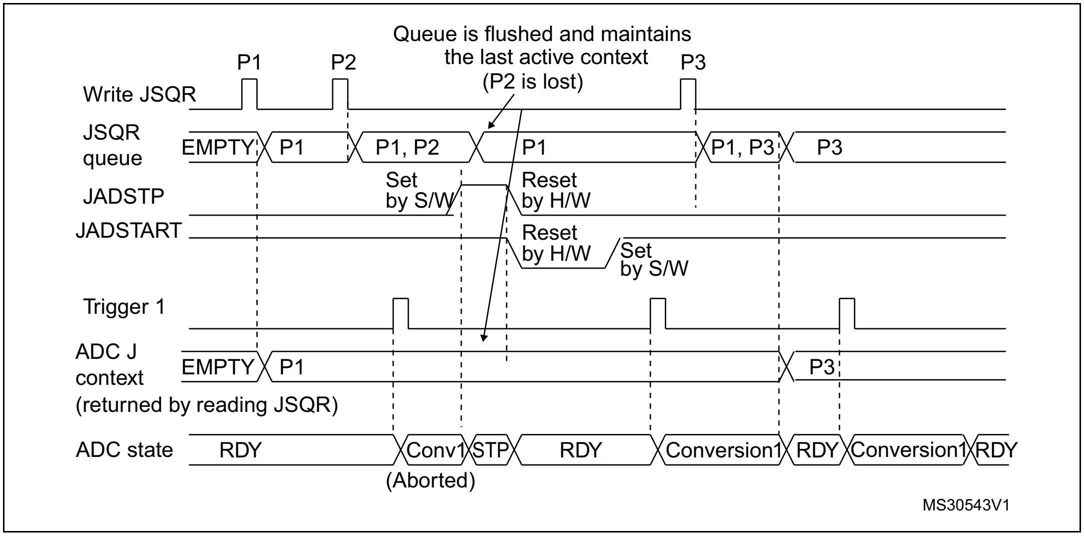
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 105. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion.



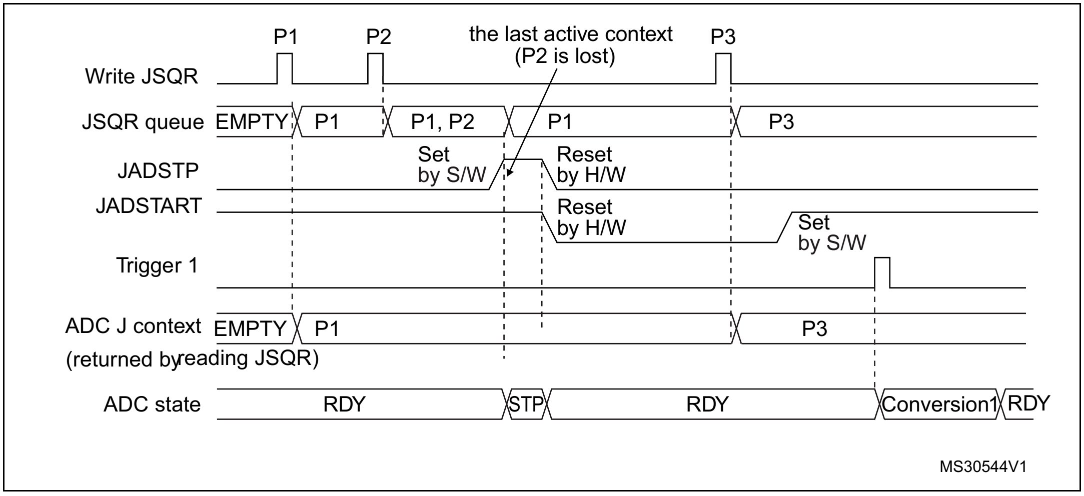
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 106. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.



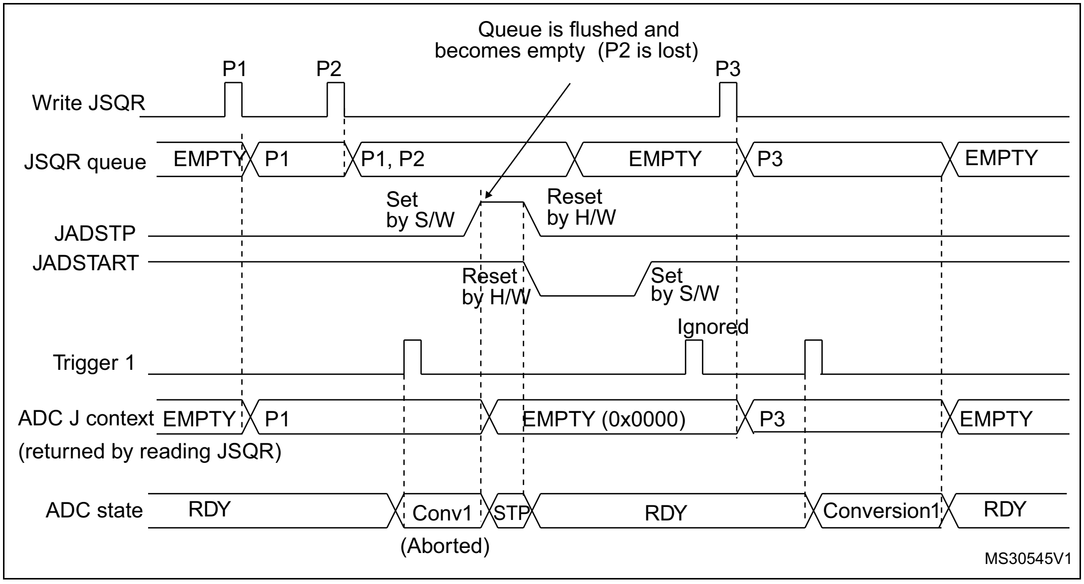
1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 107. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion



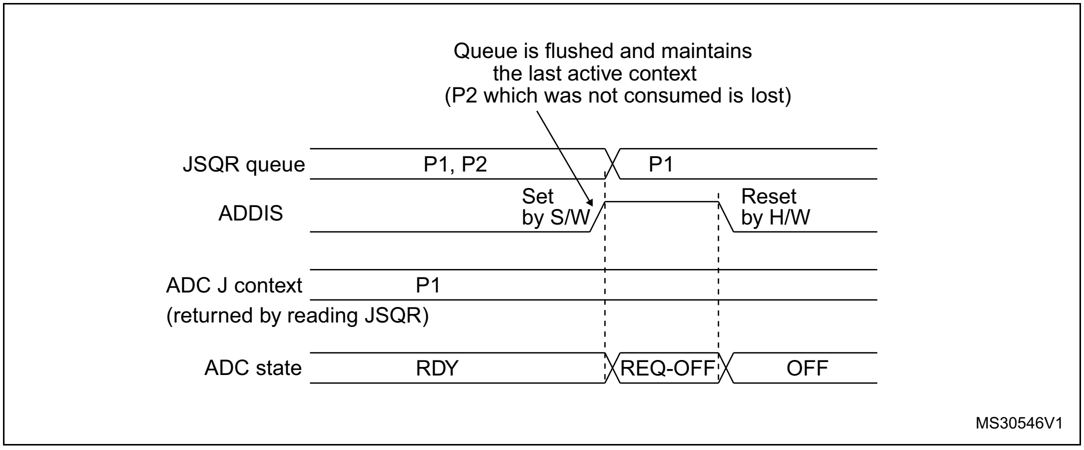
1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 108. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1)



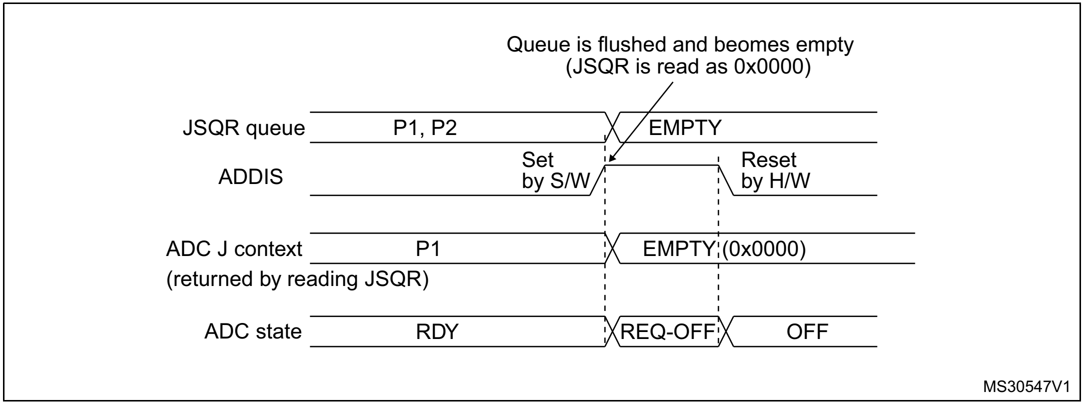
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 109. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0)


Queue is flushed and maintains the last active context (P2 which was not consumed is lost)
JSQR queue
P1, P2
P1
ADDIS
Set by S/W
Reset by H/W
ADC J context
(returned by reading JSQR)
P1
ADC state
RDY
REQ-OFF
OFF
MS30546V1

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 110. Flushing JSQR queue of context by setting ADDIS=1 (JQM=1)



1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN[1:0] not equal to 00 (otherwise triggering a software conversion)
6. Set JADSTART
7. Set JADSTP
8. Wait until JADSTART is reset
9. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS=1 into the ADC_CFGR register.

21.4.22 Programmable resolution (RES) - Fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. *Figure 115*, *Figure 116*, *Figure 117* and *Figure 118* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 167*.

Table 167. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at F_{ADC}= 30 MHz	T_{CONV} (ADC clock cycles) (with Sampling Time= 2.5 ADC clock cycles)	T_{CONV} (ns) at F_{ADC}= 30 MHz
12	12.5 ADC clock cycles	416.67 ns	15 ADC clock cycles	500.0 ns
10	10.5 ADC clock cycles	350.0 ns	13 ADC clock cycles	433.33 ns
8	8.5 ADC clock cycles	203.33 ns	11 ADC clock cycles	366.67 ns
6	6.5 ADC clock cycles	216.67 ns	9 ADC clock cycles	300.0 ns

21.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

21.4.24 End of conversion sequence (EOS, JEOS)

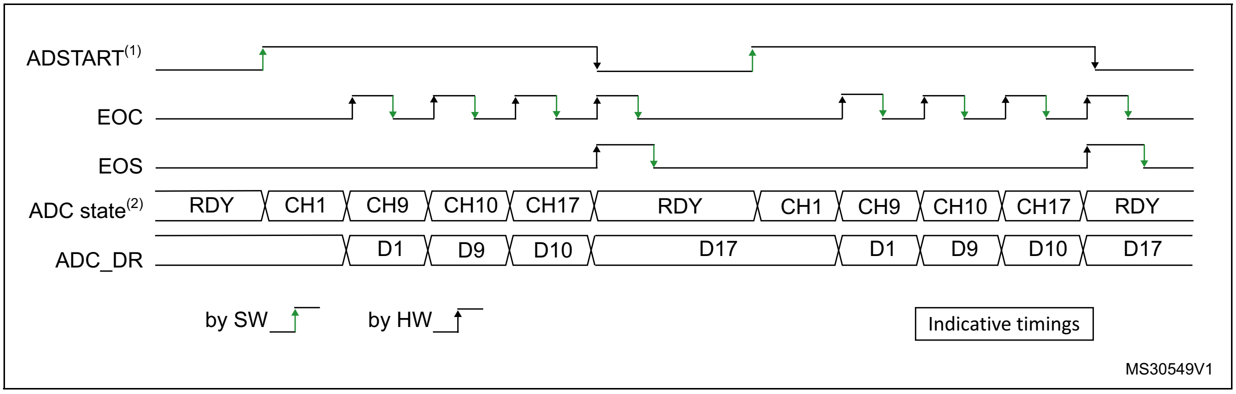
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

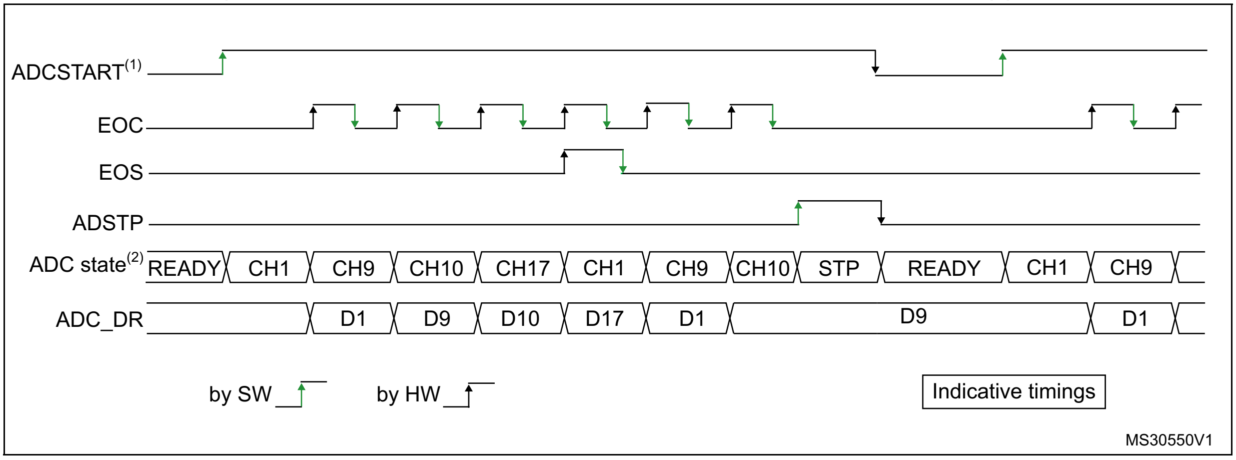
21.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 111. Single conversions of a sequence, software trigger


ADSTART(1)
EOC
EOS
ADC state(2)
RDY CH1 CH9 CH10 CH17 RDY CH1 CH9 CH10 CH17 RDY
ADC_DR
D1 D9 D10 D17 D1 D9 D10 D17
by SW
by HW
Indicative timings
MS30549V1

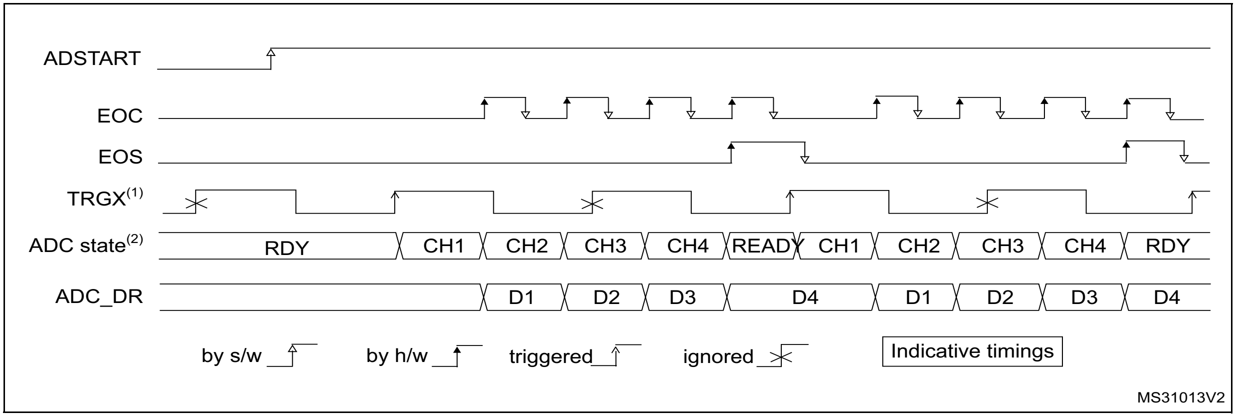
1. EXTEN[1:0]=00, CONT=0
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 112. Continuous conversion of a sequence, software trigger



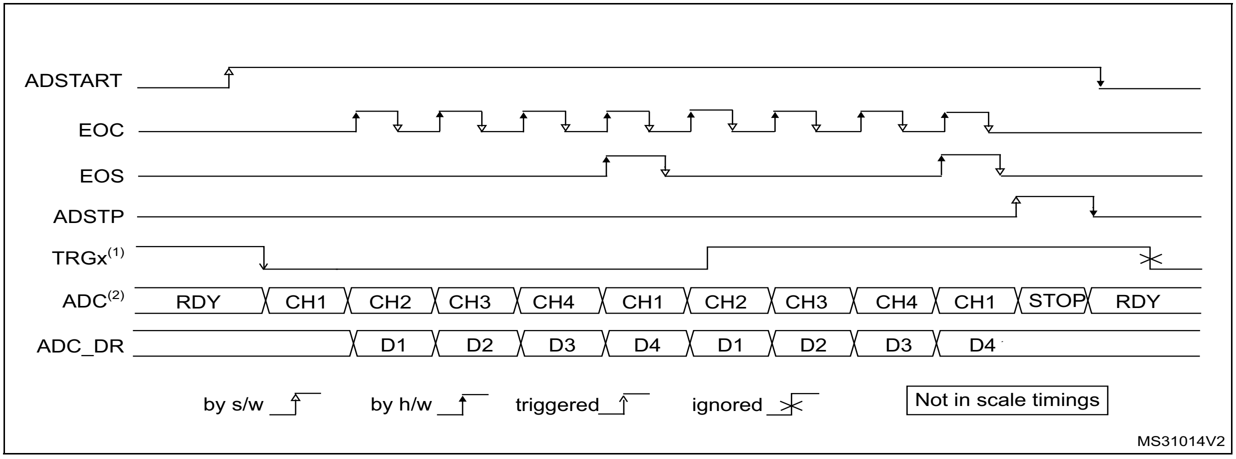
1. EXTEN[1:0]=00, CONT=1
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 113. Single conversions of a sequence, hardware trigger



1. TRGx (over-frequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

Figure 114. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

21.4.26 Data management

Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADC_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 115*, *Figure 116*, *Figure 117* and *Figure 118*.

Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. In that case, the data are aligned on a byte basis as shown in *Figure 117* and *Figure 118*.

Note: Left-alignment is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the ALIGN bit value is ignored and the ADC only provides right-aligned data.

Offset

An offset y (y=1,2,3,4) can be applied to a channel by setting the bit OFFSETy_EN=1 into ADC_OFRy register. The channel to which the offset will be applied is programmed into the bits OFFSETy_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSETy[11:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

Note: Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy_EN bit in ADC_OFRy register is ignored (considered as reset).

Table 170 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 168. Offset computation versus data resolution

Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed 12-bit data	-
01: 10-bit	DATA[11:2],00	OFFSET[11:0]	Signed 10-bit data	The user must configure OFFSET[1:0] to “00”

Table 168. Offset computation versus data resolution (continued)

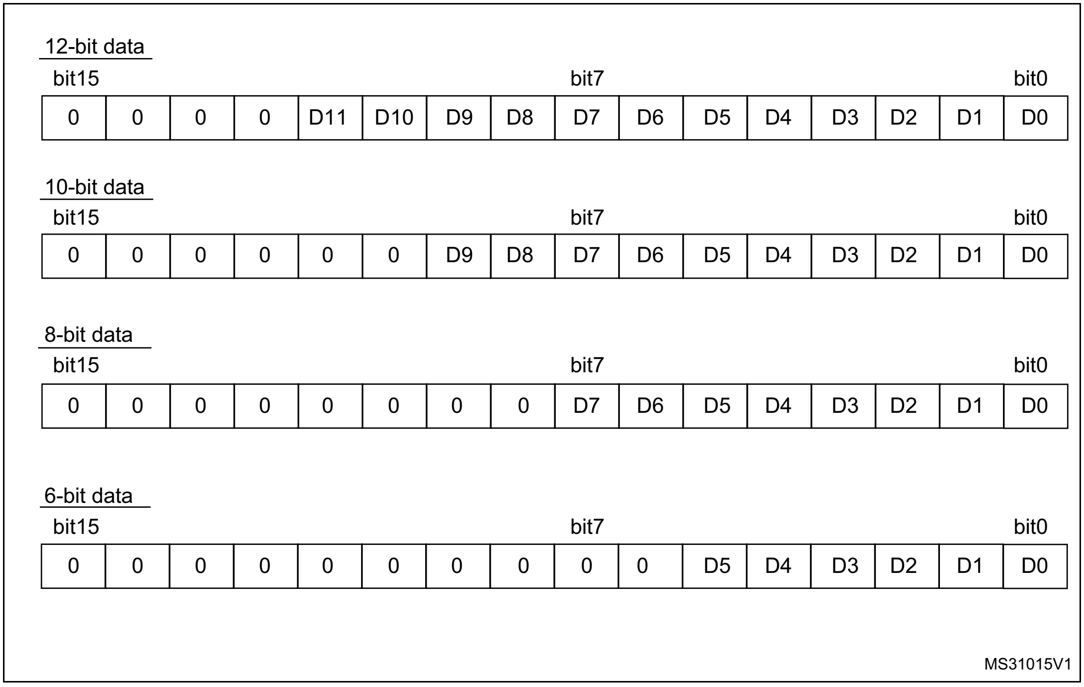
Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
10: 8-bit	DATA[11:4],0000	OFFSET[11:0]	Signed 8-bit data	The user must configure OFFSET[3:0] to “0000”
11: 6-bit	DATA[11:6],000000	OFFSET[11:0]	Signed 6-bit data	The user must configure OFFSET[5:0] to “000000”

When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y=1,2,3,4) corresponding to the channel “i”:

- If one of the offsets is enabled (bit OFFSETy_EN=1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

Figure 115, *Figure 116*, *Figure 117* and *Figure 118* show alignments for signed and unsigned data.

Figure 115. Right alignment (offset disabled, unsigned value)



12-bit data

bit15								bit7							bit0
0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

10-bit data

bit15								bit7							bit0
0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

8-bit data

bit15								bit7							bit0
0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0

6-bit data

bit15								bit7							bit0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0

MS31015V1

Figure 116. Right alignment (offset enabled, signed value)

12-bit data

bit15								bit7							bit0
SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

10-bit data

bit15								bit7							bit0
SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

8-bit data

bit15								bit7							bit0
SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	D7	D6	D5	D4	D3	D2	D1	D0

6-bit data

bit15								bit7							bit0
SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	D5	D4	D3	D2	D1	D0

MS31016V1

Figure 117. Left alignment (offset disabled, unsigned value)

12-bit data

bit15								bit7							bit0
D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

10-bit data

bit15								bit7							bit0
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0

8-bit data

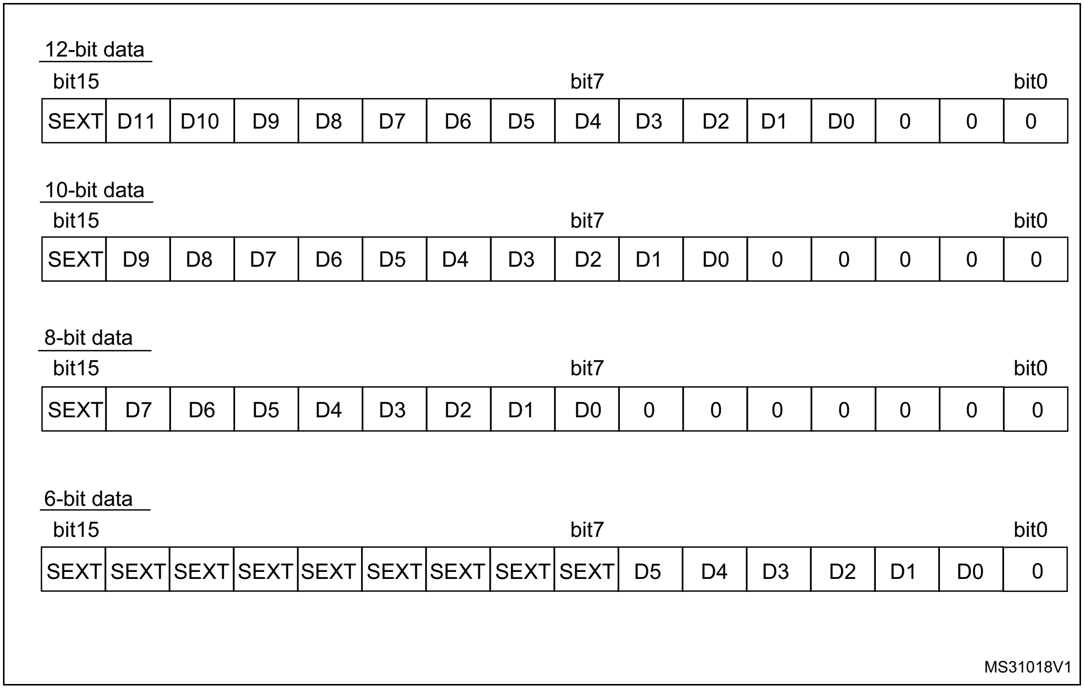
bit15								bit7							bit0
D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0	0

6-bit data

bit15								bit7							bit0
0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0	0	0

MS31017V1

Figure 118. Left alignment (offset enabled, signed value)



12-bit data

bit15								bit7							bit0
SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0

10-bit data

bit15								bit7							bit0
SEXT	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0

8-bit data

bit15								bit7							bit0
SEXT	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0

6-bit data

bit15								bit7							bit0
SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	D5	D4	D3	D2	D1	D0	0

Gain compensation

When GCOMP bit is set in ADC_CFGR2 register, the gain compensation is activated on all the converted data. After each conversion, data is calculated with the following formula.

$$\mathrm{DATA} = \mathrm{DATA}(\text{adc result}) \times (\mathrm{GCOMPCOEFF}) / 4096$$

As GCOMPCOEFF can be programmed from 0 to 16383, the actual gain compensation factor can range from 0 to 3.999756.

Before storing the resulting data in RDATA or JDATAx registers, the LSB−1 value is evaluated to round up the data and minimize the error.

The gain compensation is also effective for the oversampling. When the gain compensation is used for the oversampling mode, the gain calculation is performed after the accumulation and right-shift operations to minimize the power consumption (the gain calculation is done only once instead of at each conversion).

Offset compensation

When SATEN bit is set in ADC_OFRy register during offset operation, data are unsigned. All the offset data saturate at 0x000 (in 12-bit mode). When OFFSETPOS bit is set, the offset direction is positive and the data saturate at 0xFFF (in 12-bit mode). In 8-bit mode, data saturate at 0x00 and 0xFF, respectively.

The analog watchdog comparison is performed on unsigned values, after offset and gain compensation. For correct watchdog operation, the data after offset compensation must be in unsigned format (SATEN bit set to 1 in ADC_OFRy register).

ADC overrun (OVR, OVRMOD)

The overrun flag (OSR) notifies of that a buffer overrun event occurred when the regular converted data has not been read (by the CPU or the DMA) before new converted data became available.

The OVR flag is set if the EOC flag is still 1 at the time when a new conversion completes. An interrupt can be generated if bit OVRIE=1.

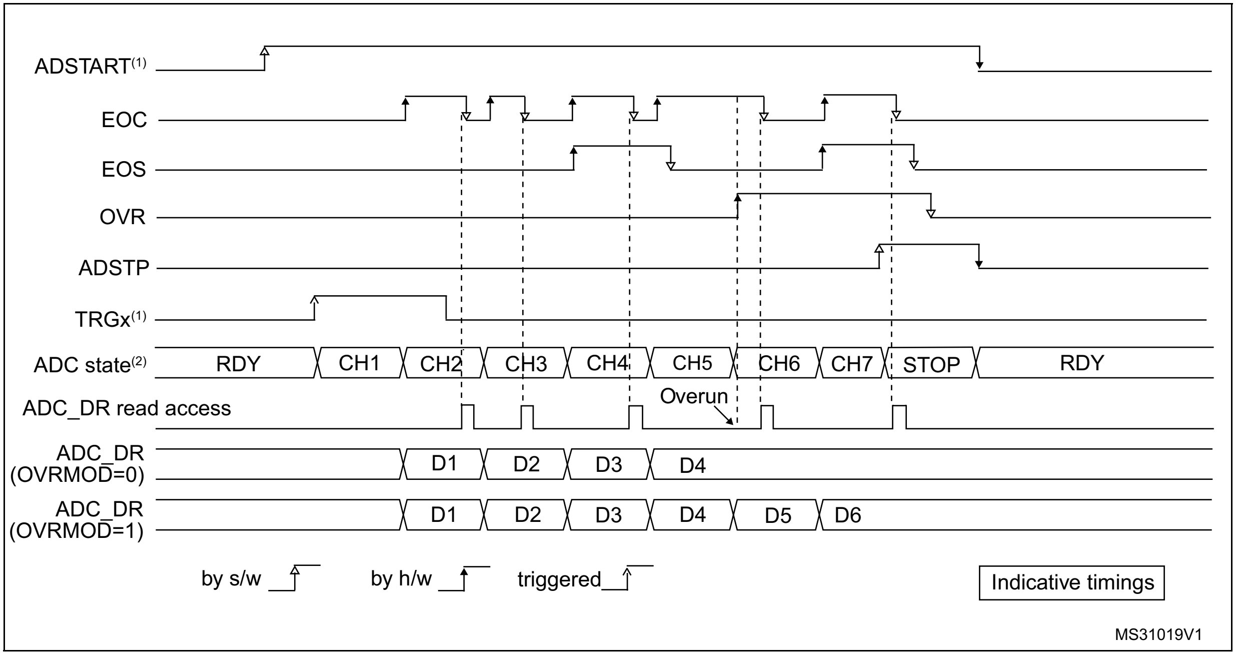
When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting bit ADSTP=1.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD=0: The overrun event preserves the data register from being overrun: the old data is maintained and the new conversion is discarded and lost. If OVR remains at 1, any further conversions will occur but the result data will be also 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, any further conversions will operate normally and the ADC_DR register will always contain the latest converted data.

Figure 119. Example of overrun (OVR)



Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD should be configured to 0 to manage overrun events as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event will not prevent the ADC from continuing to convert and the ADC_DR register will always contain the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

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 : ADC overrun (OVR, OVRMOD)*).

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 of the ADC_CFGR register in single ADC mode, or with bit DMACFG of the ADC_CCR register in dual ADC mode:

- DMA one shot mode (DMACFG=0).
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMACFG=1)
  This mode is suitable when programming the DMA in circular mode.

DMA one shot mode (DMACFG=0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when DMA_EOT interrupt occurs - refer to DMA paragraph) 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 with 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.
- 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 is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

21.4.27 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY=1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see *Figure 120*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 121*).

This is a way to automatically adapt the speed of the ADC to the speed of the system which will read the data.

The delay is inserted after each regular conversion (whatever DISCEN=0 or 1) and after each sequence of injected conversions (whatever JDISCEN=0 or 1).

Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

Note: This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to restart a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see *Figure 121*).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see *Figure 123*).

The behavior is slightly different in auto-injected mode (JAUTO=1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 124*).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO=1, CONT=1 and AUTDLY=1), follow the following procedure:

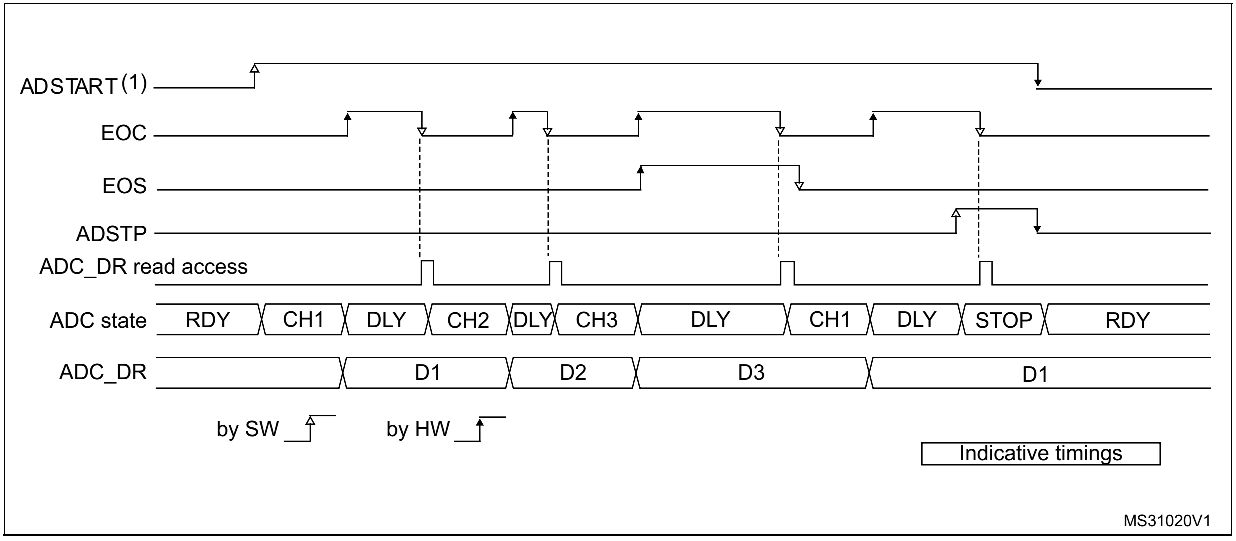
1. Wait until JEOS=1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP=1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can restart if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

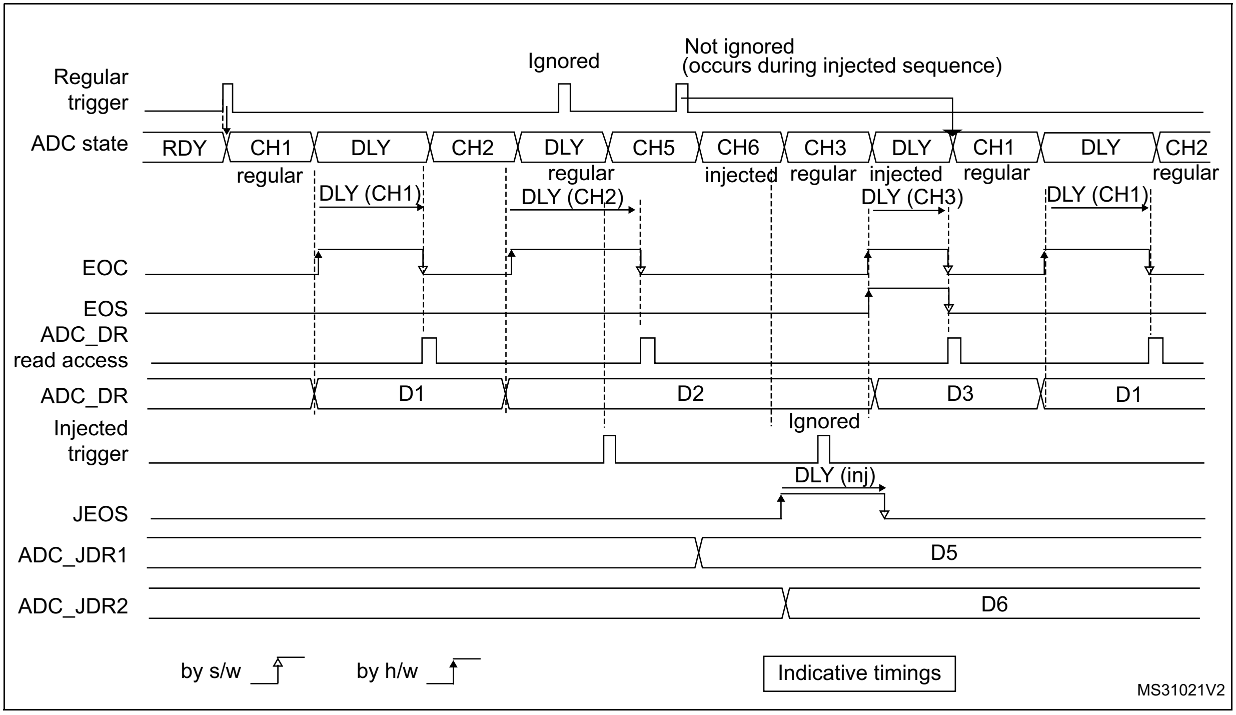
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 120. AUTODLY=1, regular conversion in continuous mode, software trigger



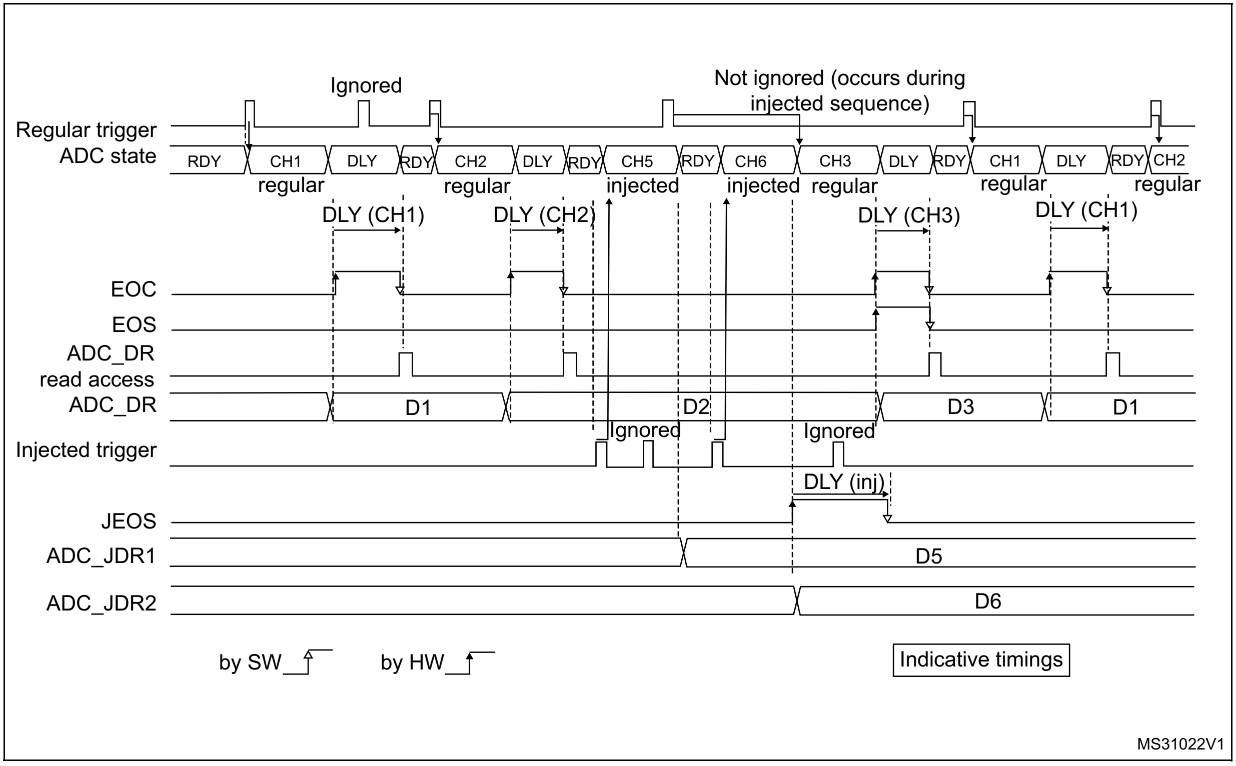
1. AUTDLY=1
2. Regular configuration: EXTEN[1:0]=00 (SW trigger), CONT=1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

Figure 121. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)



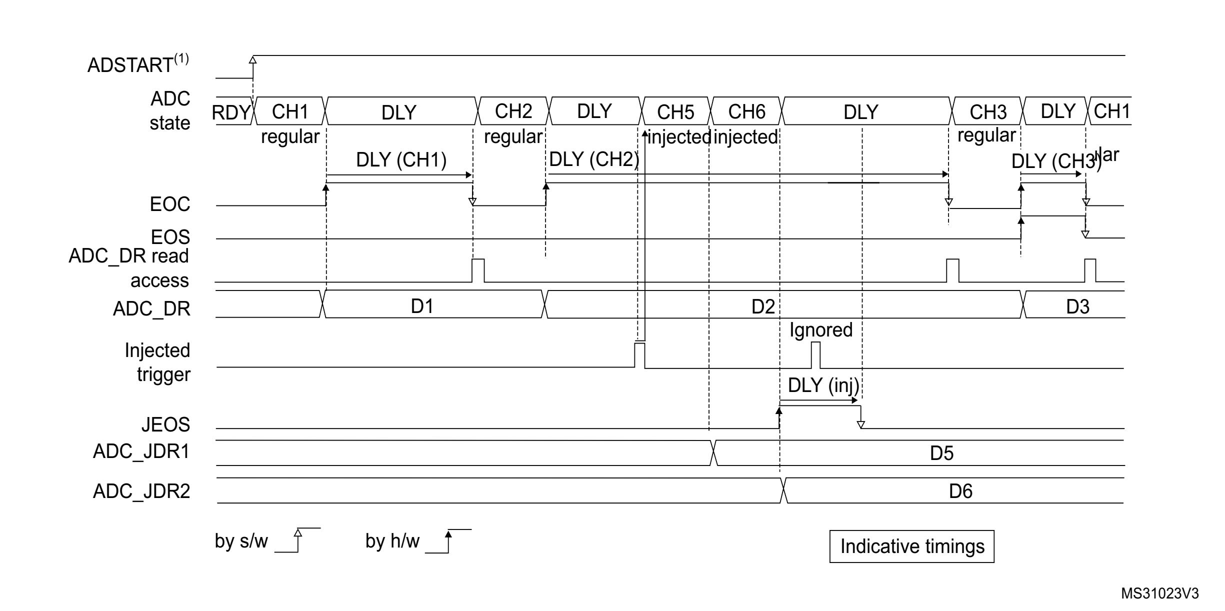
1. AUTDLY=1
2. Regular configuration: EXTEN[1:0]=01 (HW trigger), CONT=0, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN[1:0]=01 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 122. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)



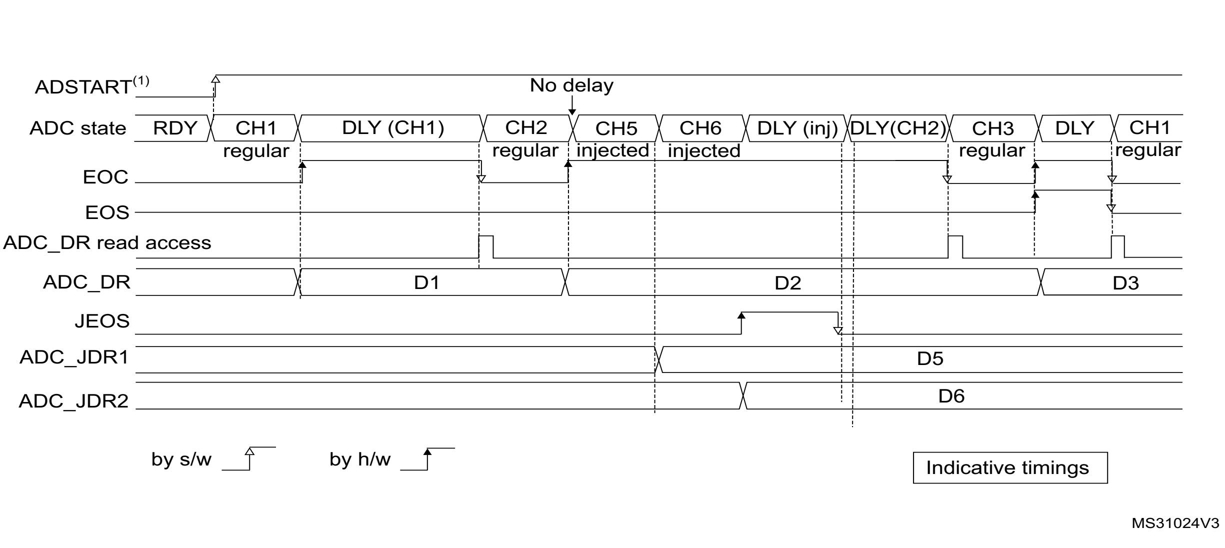
1. AUTDLY=1
2. Regular configuration: EXTEN[1:0]=01 (HW trigger), CONT=0, DISCEN=1, DISCNUM=1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0]=01 (HW Trigger), JDISCEN=1, CHANNELS = 5,6

Figure 123. AUTODLY=1, regular continuous conversions interrupted by injected conversions



1. AUTDLY=1
2. Regular configuration: EXTEN[1:0]=00 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN[1:0]=01 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 124. AUTODLY=1 in auto- injected mode (JAUTO=1)

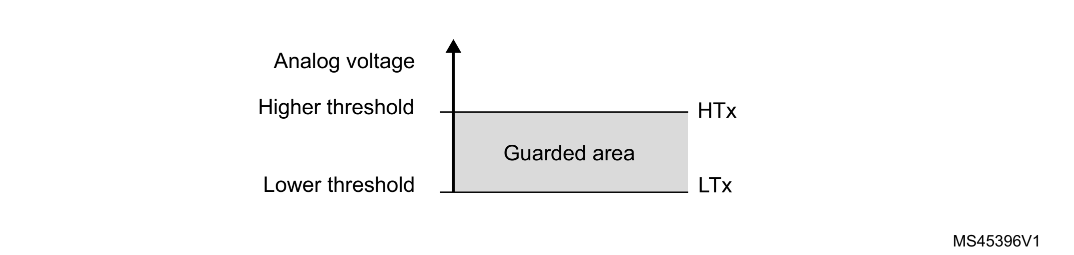


1. AUTDLY=1
2. Regular configuration: EXTEN[1:0]=00 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2
3. Injected configuration: JAUTO=1, CHANNELS = 5,6

21.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 125. Analog watchdog guarded area



AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADC_IER register (x=1,2,3).

AWDx (x=1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels[(1)] remain within a configured voltage range (window).

Table 169 shows how the ADC_CFGR registers should be configured to enable the analog watchdog on one or more channels.

Table 169. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single[(1)] injected channel	1	0	1
Single[(1)] regular channel	1	1	0
Single[(1)] regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

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 bits HT1[11:0] and LT1[11:0] of the ADC_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[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 170 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 170. Analog watchdog 1 comparison

Resolution(bit RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned[1]	Thresholds	
00: 12-bit	DATA[11:0]	LT1[11:0] and HT1[11:0]	-
01: 10-bit	DATA[11:2],00	LT1[11:0] and HT1[11:0]	User must configure LT1[1:0] and HT1[1:0] to 00
10: 8-bit	DATA[11:4],0000	LT1[11:0] and HT1[11:0]	User must configure LT1[3:0] and HT1[3:0] to 0000
11: 6-bit	DATA[11:6],000000	LT1[11:0] and HT1[11:0]	User must configure LT1[5:0] and HT1[5:0] to 000000

1. Refer to *Section : Gain compensation* for additional details on analog watchdog comparison.

Analog watchdog filter for watchdog 1

When an ADC is configured with only one input channel (selecting several channels in Scan mode not allowed), a valid ADC conversion data interval can be configured through the ADC_TR1 register:

- When converted data belong to the interval defined in ADC_TR1, a DMA request is generated.
- Otherwise, no DMA request is issued. RDATA register is updated at each conversion. If data are out-of-range a number of times higher than the value specified in AWDFILT bit of ADC_TR1, the AWDx flag is set an the corresponding interrupt is issued.

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 corresponding bits in AWDxCH[18:0] (x=2,3).

The corresponding watchdog is enabled when any bit of AWDxCH[18:0] (x=2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. *Table 171* describes how the comparison is performed for all the possible resolutions.

Table 171. Analog watchdog 2 and 3 comparison

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned[1]	Thresholds	
00: 12-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:0] are not relevant for the comparison
01: 10-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:2] are not relevant for the comparison
10: 8-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	-
11: 6-bit	DATA[11:6],00	LTx[7:0] and HTx[7:0]	User must configure LTx[1:0] and HTx[1:0] to 00

1. Refer to *Section : Gain compensation* for additional details on analog watchdog comparison.

ADCy_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y=ADC number, x=watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_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).
- ADCy_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS=1). Note that stopping regular or injected conversions (setting ADSTP=1 or JADSTP=1) has no influence on the generation of ADCy_AWDx_OUT.

Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 126. ADCy_AWDx_OUT signal generation (on all regular channels)

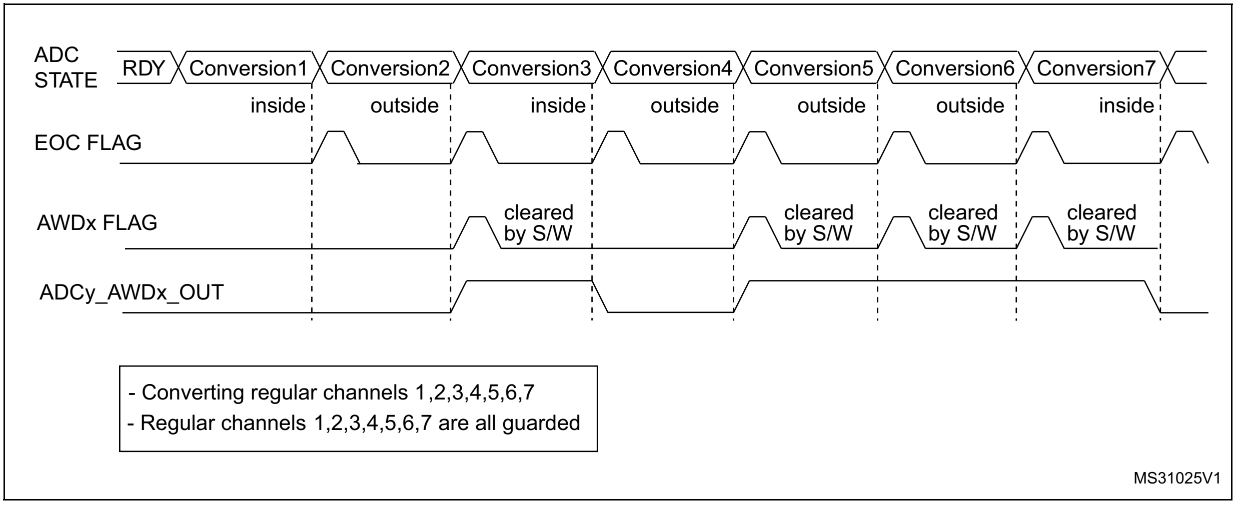


Figure 127. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software)

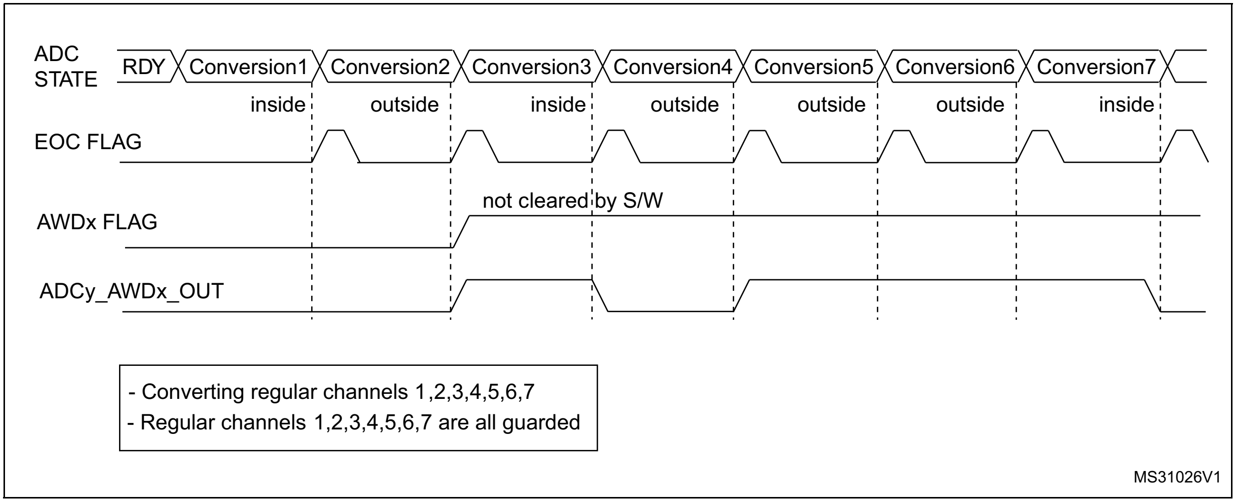


Figure 128. ADCy_AWDx_OUT signal generation (on a single regular channel)

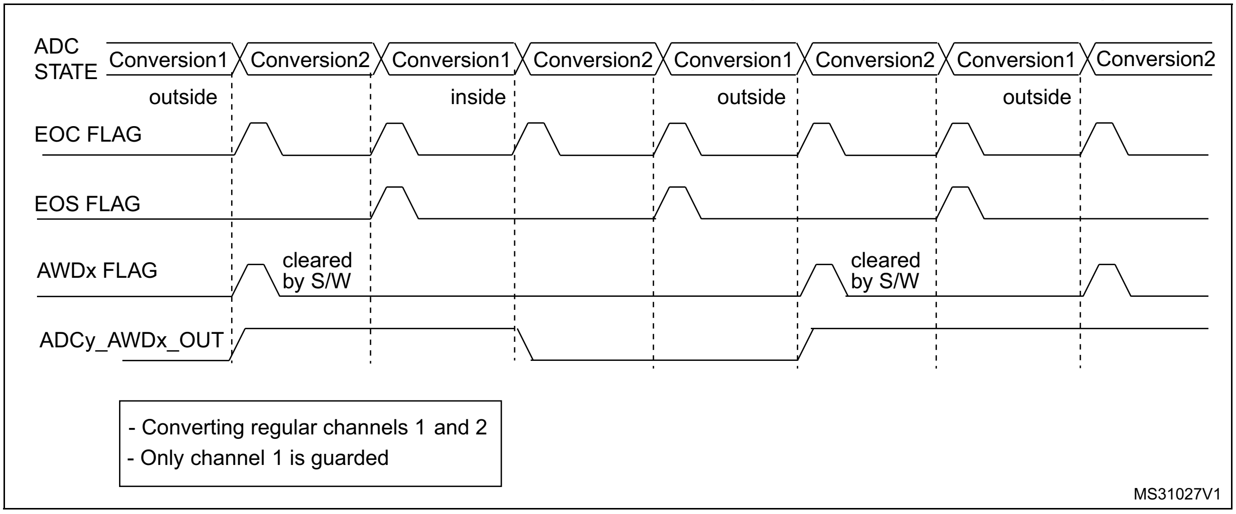
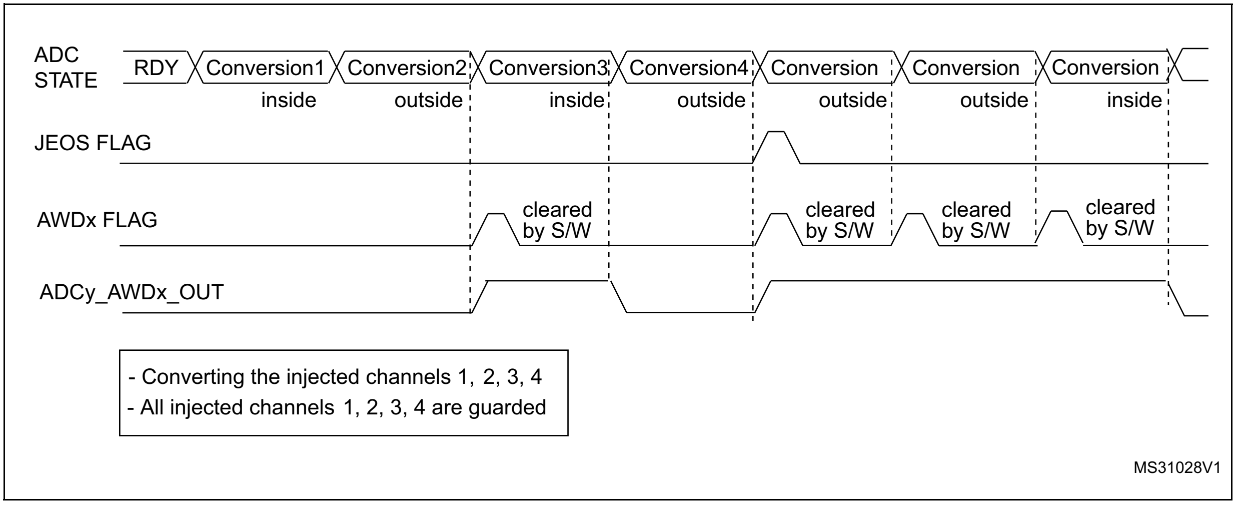


Figure 129. ADCy_AWDx_OUT signal generation (on all injected channels)



Analog watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTx[11:0] and HTx[11:0] are updated during the ADC conversion of the ADC guarded channel, the watchdog function is masked for this conversion. This masking will be removed at the next start of conversion, resulting in a analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new interval, no interrupt and AWDx_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion which started after the threshold update. If AWD_xOUT is already asserted, programming the new thresholds does not deassert the AWDx_OUT signal.

Analog watchdog with gain and offset compensation

When gain and offset compensation are enabled, the analog watchdog compares the threshold after the compensated data.

Note: When the offset compensation is enabled (OFFSETy_EN set to 1 in ADC_OFRy register), data overflow or underflow can result in a wrong watchdog result. When the saturation is enabled (SATEN set to 1 in ADC_OFRy), the watchdog provides a correct result. However this prevents from using the signed data format.

21.4.29 Oversampler

The oversampling unit performs data pre-processing to offload the CPU. It is able to 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 to perform by hardware the following functions: 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, and can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to 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 intermediary result after the shifting exceeds 16-bit, the result is truncated as is, without saturation.

Figure 130. 20-bit to 16-bit result truncation

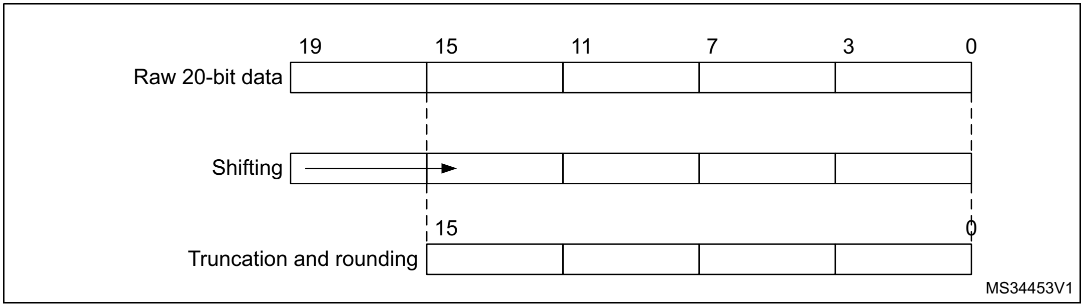


Figure 131 gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 131. Numerical example with 5-bit shift and rounding

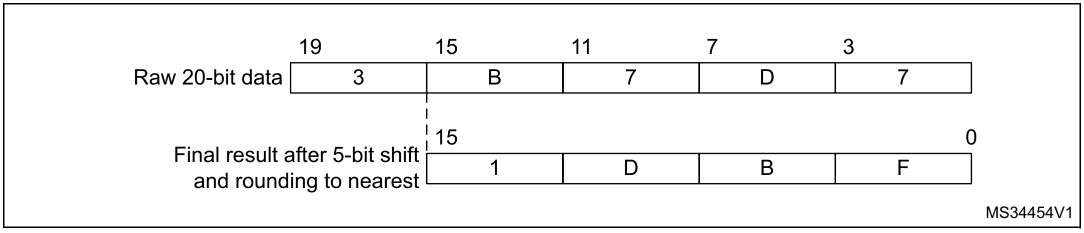


Table 172 gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 172. Maximum output results versus N and M (gray cells indicate truncation)

Over sampling 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	0x020
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

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversions, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follows:

- 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 (EOS) occurs once the sequence of oversampled data is completed (i.e. after N x sequence length conversions total)

ADC operating modes supported when oversampling (single ADC mode)

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous mode conversions
- 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 (AUTDLY)
- 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.

Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy_EN bit in ADC_OFRy register is ignored (considered as reset).

Analog watchdog

The analog watchdog functionality is maintained (AWDSGL and AWDEN bits), with the following difference:

- The RES[1:0] bits are ignored, comparison is always done using the full 12-bit values HT[11:0] and LT[11:0]
- the comparison is performed on the most significant 12-bit of the 16-bit oversampled results ADC_DR[15:4]

Note: Care must be taken when using high shifting values, this will reduce the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HT[0:7] / LT[[0:7], and HT[11:8] / LT[11:8] must be kept reset.

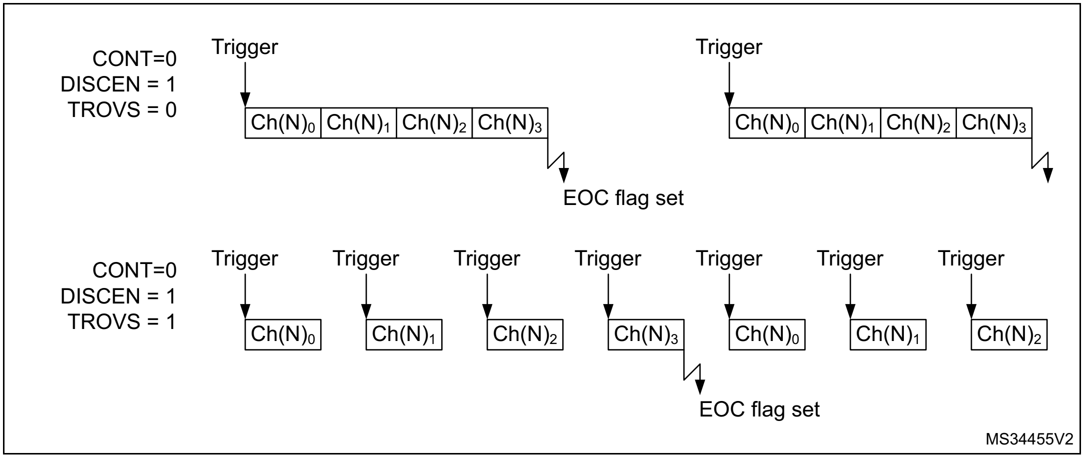
Triggered mode

The averager can also be used for basic filtering purpose. Although not a very powerful 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 TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

Figure 132 below shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 132. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

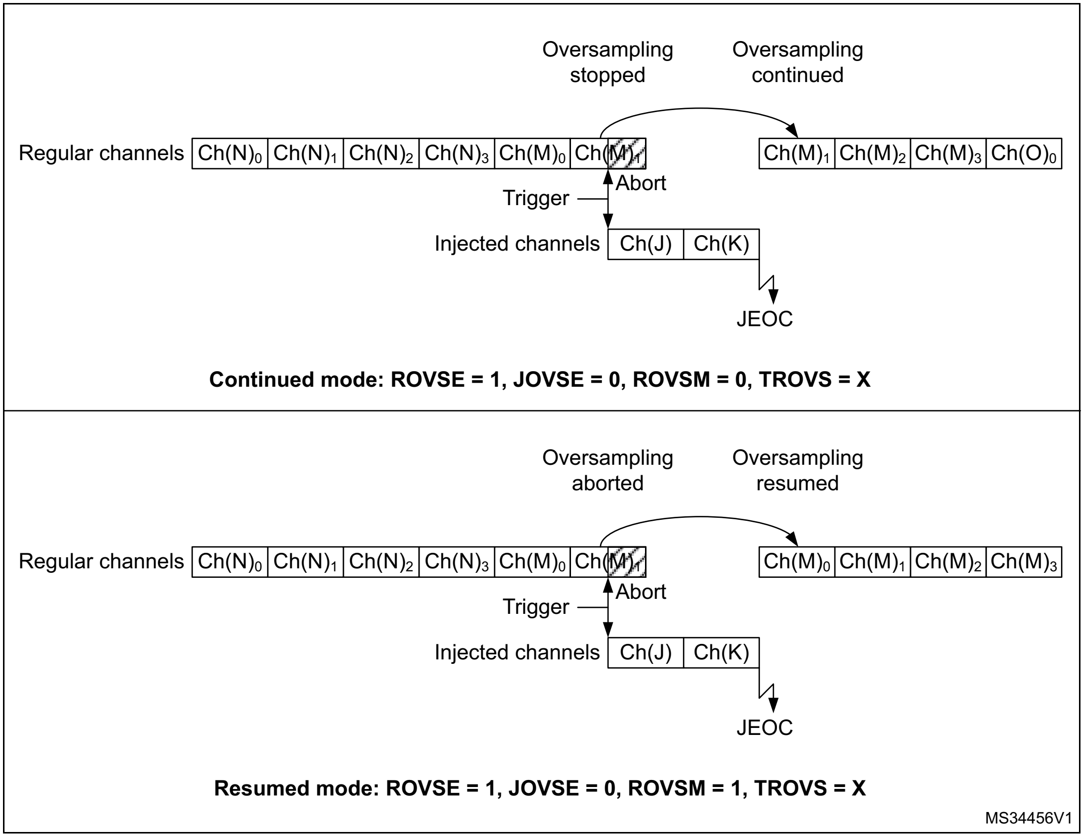
Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling will be complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);
- In resumed mode, the accumulation restarts from 0 (previous conversion results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be complete and the regular sequencer will be blocked.

Figure 133 gives examples for a 4x oversampling ratio.

Figure 133. Regular oversampling modes (4x ratio)


Oversampling stopped
Oversampling continued
Regular channels
Ch(N)0 Ch(N)1 Ch(N)2 Ch(N)3 Ch(M)0 Ch(M)1
Ch(M)1 Ch(M)2 Ch(M)3 Ch(O)0
Abort
Trigger
Injected channels
Ch(J) Ch(K)
JEOC
Continued mode: ROVSE = 1, JOVSE = 0, ROVSM = 0, TROVS = X
Oversampling aborted
Oversampling resumed
Regular channels
Ch(N)0 Ch(N)1 Ch(N)2 Ch(N)3 Ch(M)0 Ch(M)1
Ch(M)0 Ch(M)1 Ch(M)2 Ch(M)3
Abort
Trigger
Injected channels
Ch(J) Ch(K)
JEOC
Resumed mode: ROVSE = 1, JOVSE = 0, ROVSM = 1, TROVS = X
MS34456V1

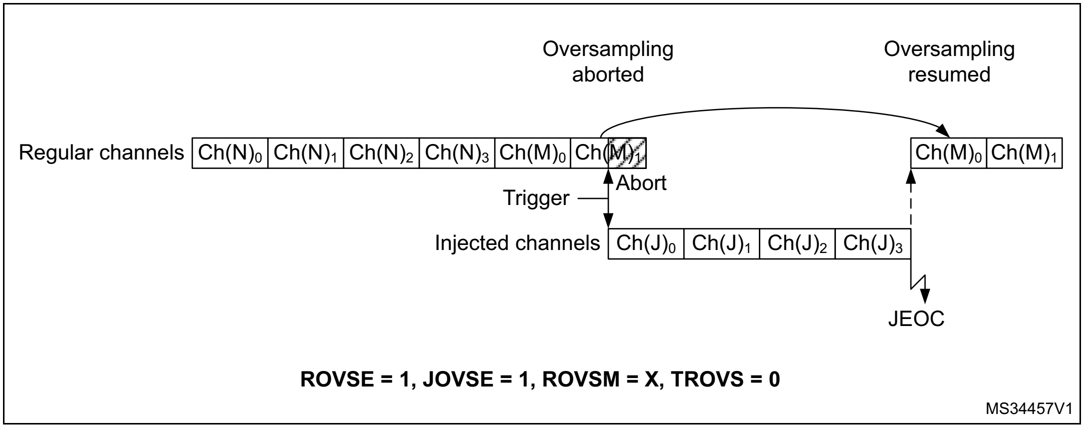
Oversampling Injected channels only

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on *Figure 134* below.

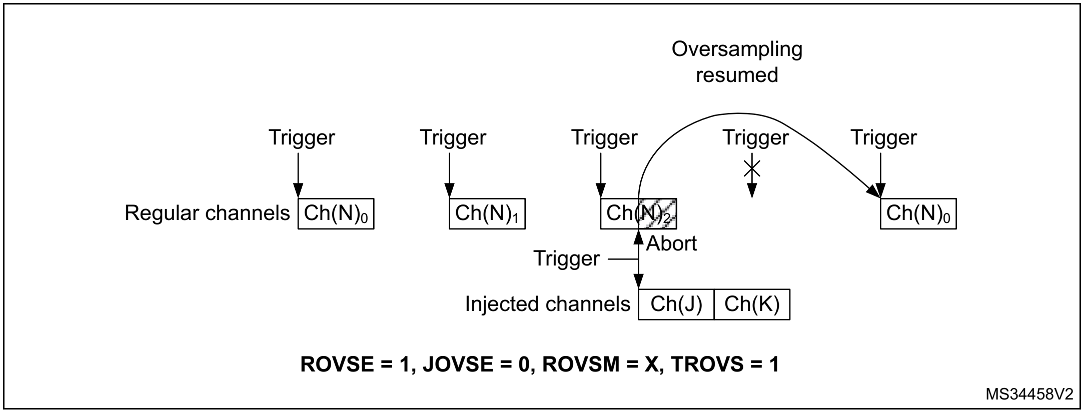
Figure 134. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on *Figure 135* below.

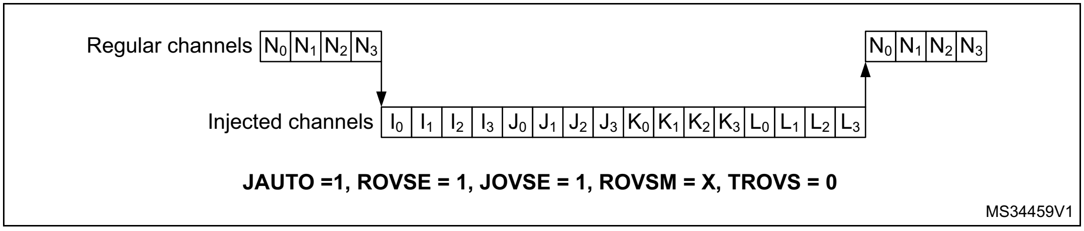
Figure 135. Triggered regular oversampling with injection



Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The *Figure 136* below shows how the conversions are sequenced.

Figure 136. Oversampling in auto-injected mode


Regular channels
N0 N1 N2 N3
N0 N1 N2 N3
Injected channels
I0 I1 I2 I3 J0 J1 J2 J3 K0 K1 K2 K3 L0 L1 L2 L3
JAUTO =1, ROVSE = 1, JOVSE = 1, ROVSM = X, TROVS = 0
MS34459V1

It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO = 1, DISCEN = 0, JDISCEN = 0, ROVSE = 1, JOVSE = 1 and TROVSE = 1.

Dual ADC modes supported when oversampling

It is possible to have oversampling enabled when working in dual ADC configuration, for the injected simultaneous mode and regular simultaneous mode. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (ROVSE = 1 or JOVSE = 1).

Combined modes summary

The *Table 173* below summarizes all combinations, including modes not supported.

Table 173. Oversampler operating modes summary

Regular Oversampling ROVSE	Injected Oversampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered Regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

21.4.30 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see *Figure 137*).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADCx master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCx_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode
- Injected simultaneous mode + Interleaved mode

In dual ADC mode (when bits DUAL[4:0] in ADCx_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JQM, JAUTO of the ADC_CFGR register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

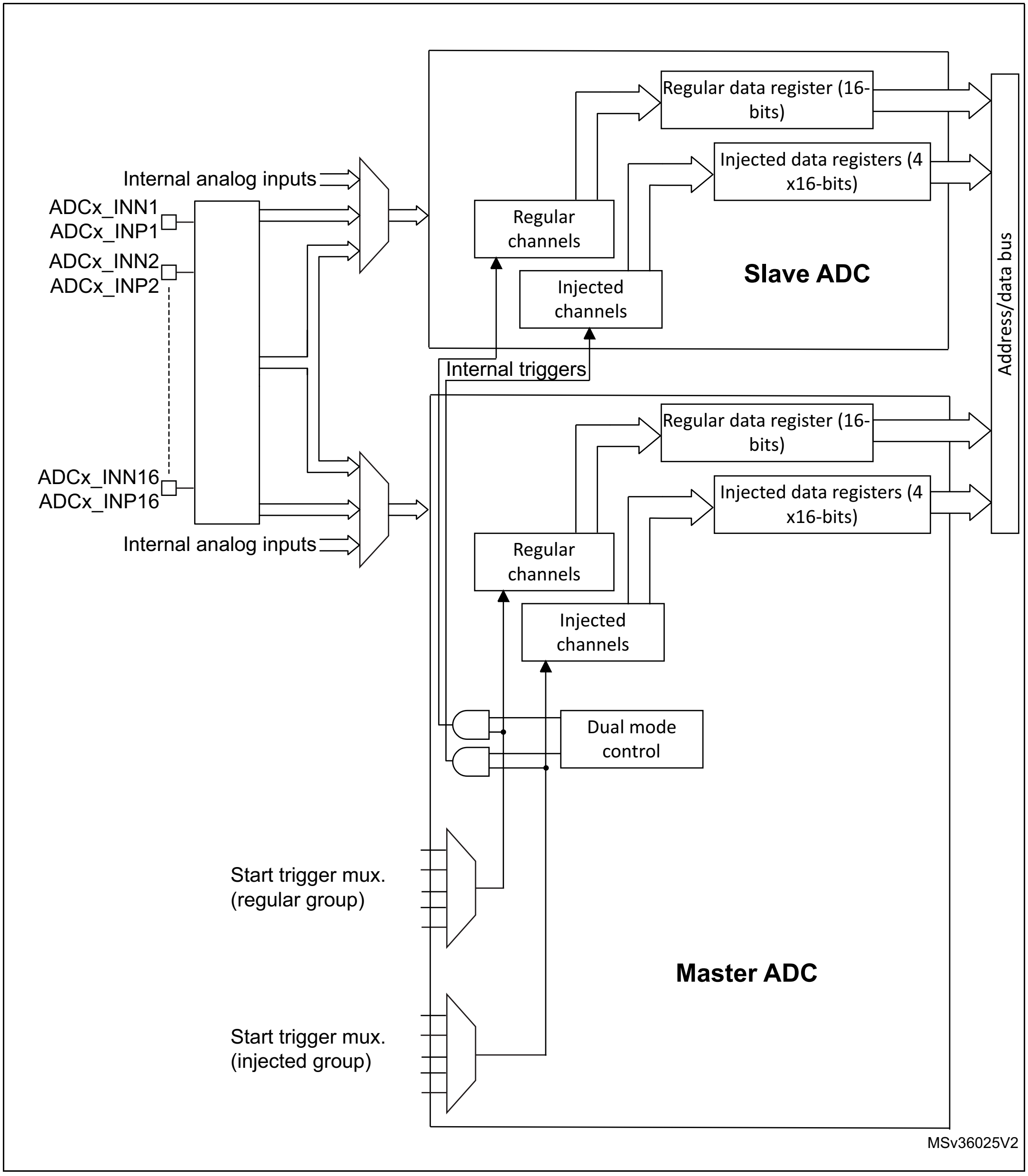
To start a conversion in dual mode, the user must program the bits EXTEN[1:0], EXTSEL, JEXTEN[1:0], JEXTSEL of the master ADC only, to configure a software or hardware trigger, and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don’t care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCx_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADCx_CSR).

Figure 137. Dual ADC block diagram[1]



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCx_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode

This mode is selected by programming bits DUAL[4:0]=00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

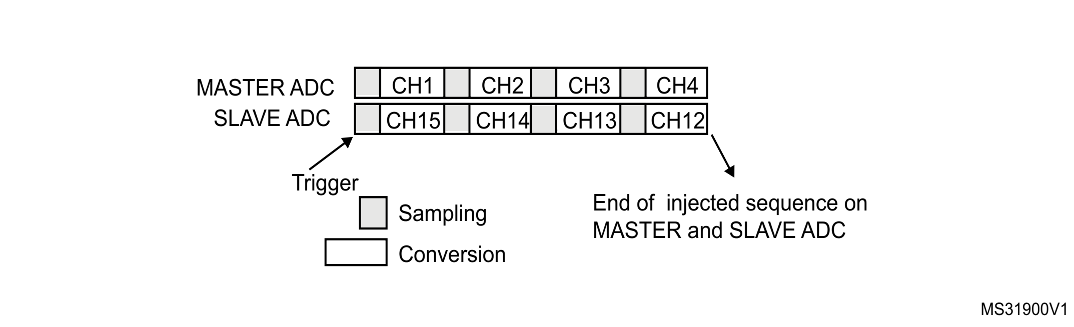
Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in *Figure 138*), it is possible for the software to enable only one of the two JEOS interrupt (ex: master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 138. Injected simultaneous mode on 4 channels: dual ADC mode



If JDISCEN=1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the

ongoing regular sequence and the associated delay phases are ignored.
There is the same behavior for regular sequences occurring on the slave ADC.

Regular simultaneous mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) will abort the current simultaneous conversions, which are restarted once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Software is notified by interrupts when it can read the data:

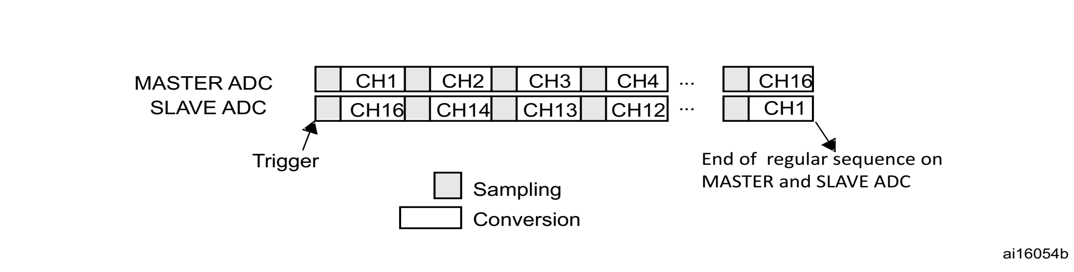
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in *Figure 139*), it is possible for the software to enable only one of the two EOC interrupt (ex: master EOC) and read both converted data from the Common Data register (ADCx_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using two DMA channels (one for the master and one for the slave). In this case bits MDMA[1:0] must be kept cleared.
  - Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  - Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Using MDMA mode, which leaves one DMA channel free for other uses:
  - Configure MDMA[1:0]=0b10 or 0b11 (depending on resolution).
  - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR)
  - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
  - Both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Note: In MDMA mode (MDMA[1:0]=0b10 or 0b11), the user must program the same number of conversions in the master's sequence as in the slave's sequence. Otherwise, the remaining conversions will not generate a DMA request.

Figure 139. Regular simultaneous mode on 16 channels: dual ADC mode



If DISCEN=1 then each “n” simultaneous conversions of the regular sequence require a regular trigger event to occur (“n” is defined by DISCNUM).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, ADCx_CDR (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADCx_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits MDMA must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master’s sequence is equal to the number of conversions in the slave’s.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

Note: This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

Interleaved mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00111.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates two conversions in interleaved mode is configured in the DELAY bits in the ADCx_CCR register. This delay starts counting one half cycle after the end of the sampling phase of the master conversion. This way, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Note: It is possible to enable only the EOC interrupt of the slave and read the common data register (ADCx_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts. It is recommended to use the MDMA mode.

It is also possible to have the regular data transferred by DMA. In this case, individual DMA requests on each ADC cannot be used and it is mandatory to use the MDMA mode, as following:

- Configure MDMA[1:0]=0b10 or 0b11 (depending on resolution).
- A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR).
- A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
- Both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Figure 140. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode

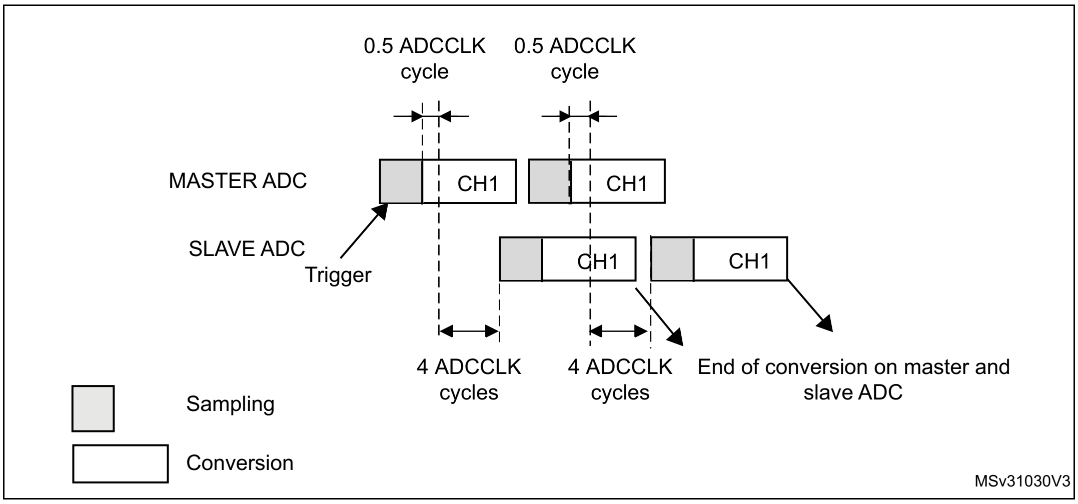
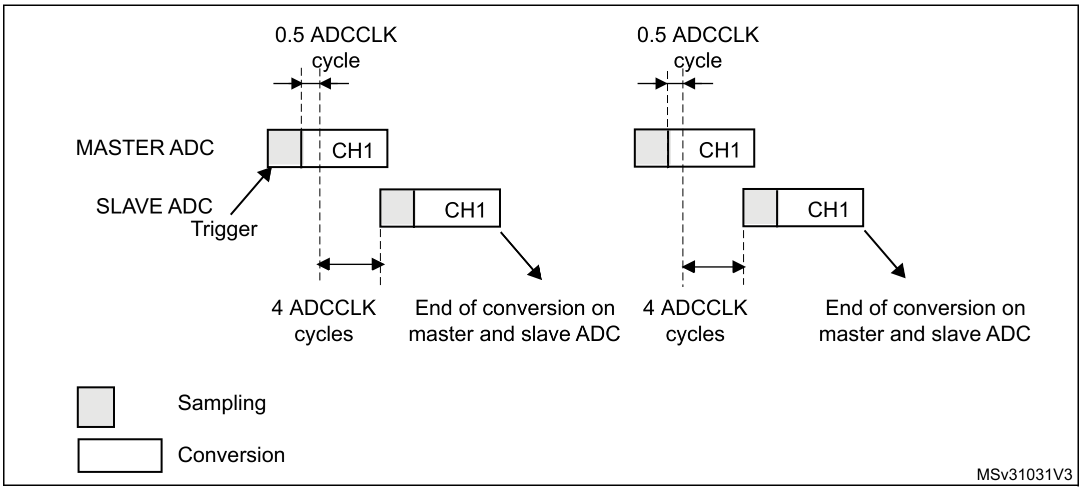


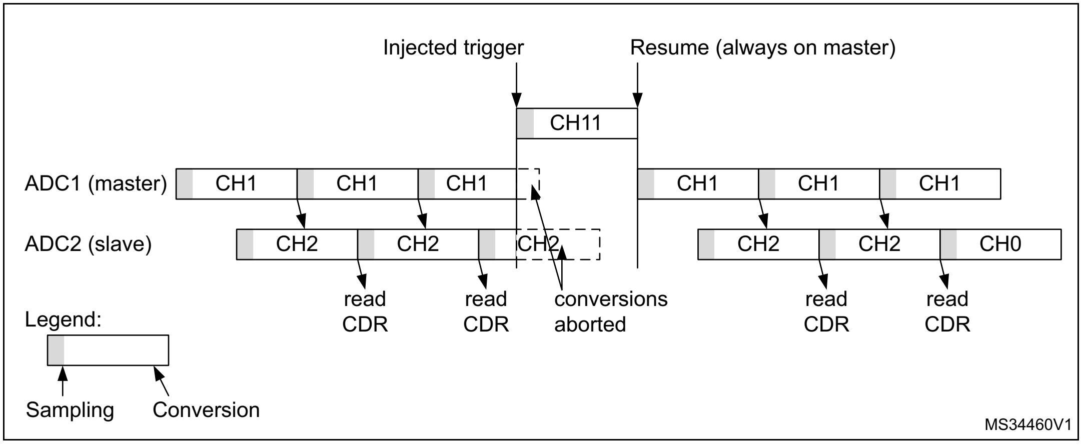
Figure 141. Interleaved mode on 1 channel in single conversion mode: dual ADC mode



If DISCEN=1, each “n” simultaneous conversions (“n” is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is restarted from the master (see *Figure 142* below).

Figure 142. Interleaved conversion with injection



Alternate trigger mode

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN[1:0] must not be 00.

Injected discontinuous mode disabled (JDISCEN=0 for both ADC)

1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

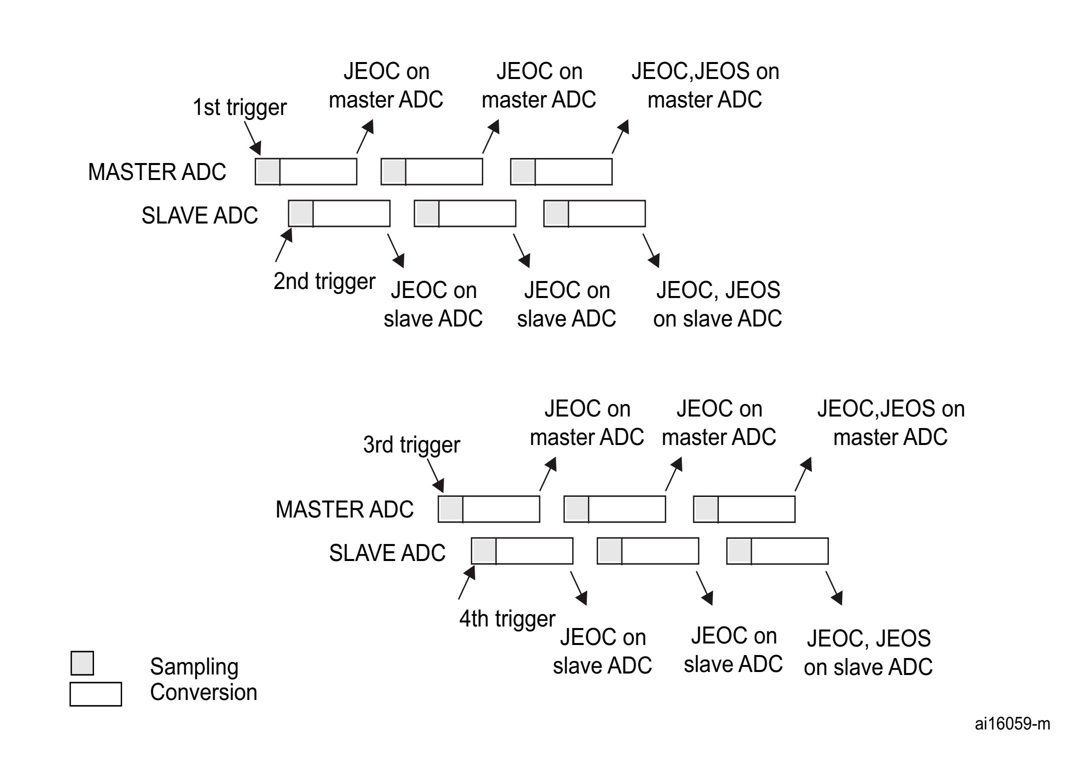
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 143. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN=1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

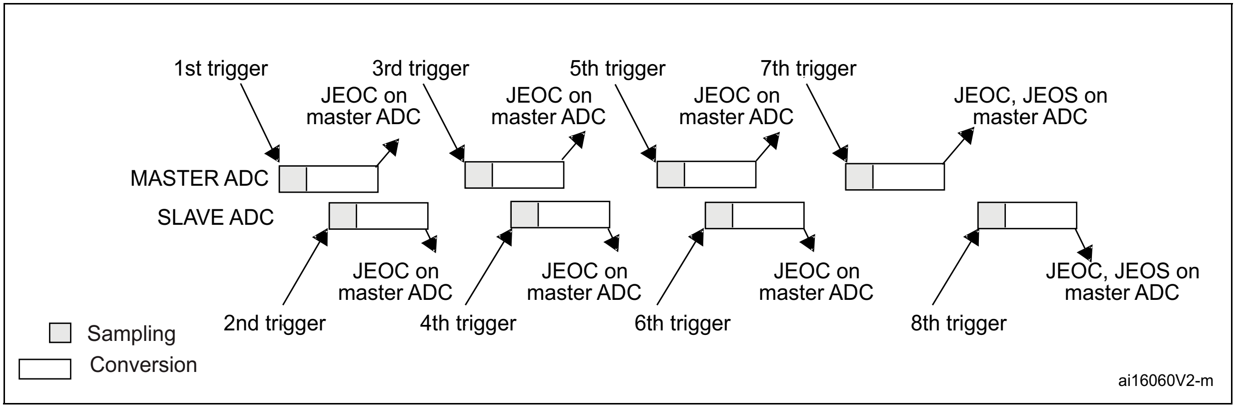
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 144. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous + alternate trigger mode

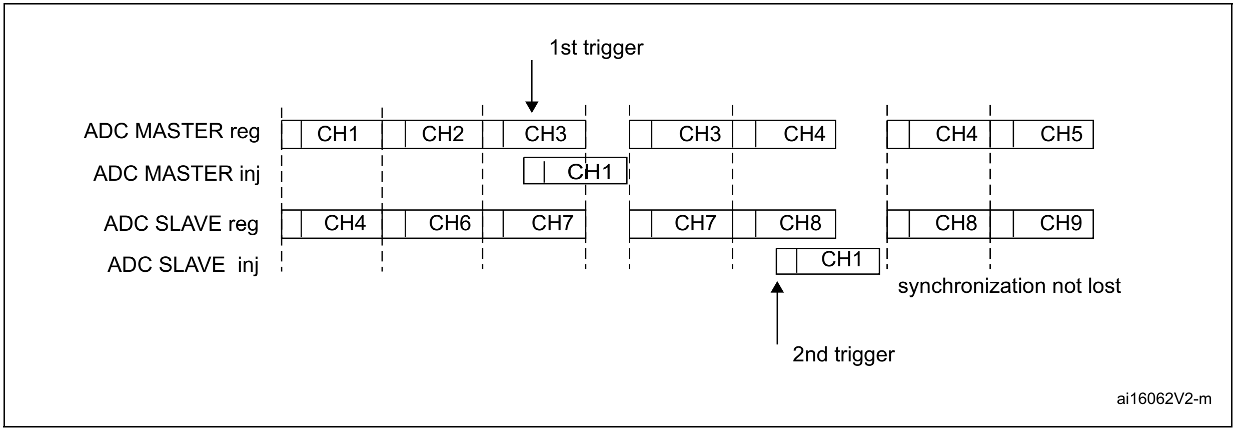
This mode is selected by programming bits DUAL[4:0]=00010.

It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. *Figure 145* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

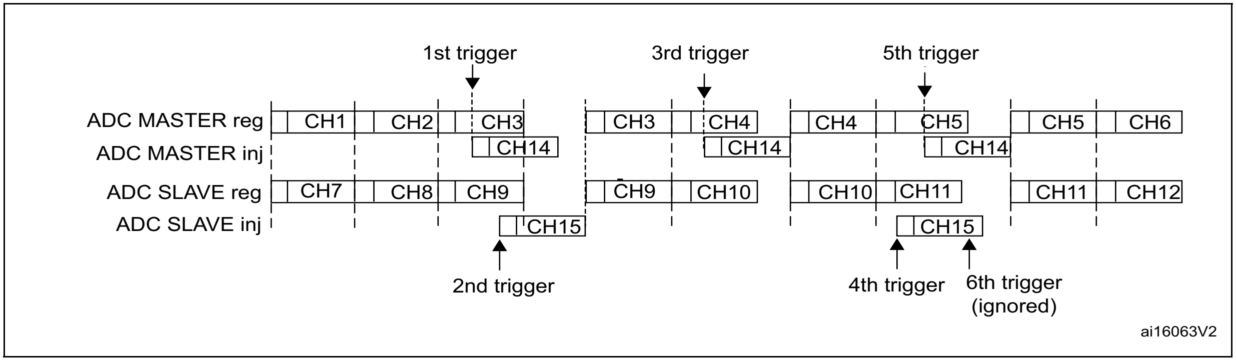
Note: In combined regular simultaneous + alternate trigger mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Figure 145. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. Figure 146 shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

Figure 146. Case of trigger occurring during injected conversion



Combined injected simultaneous plus interleaved

This mode is selected by programming bits DUAL[4:0]=00011

It is possible to interrupt an interleaved conversion with a simultaneous injected event.

In this case the interleaved conversion is interrupted immediately and the simultaneous injected conversion starts. At the end of the injected sequence the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is alway the master’s one. Figure 147, Figure 148 and Figure 149 show the behavior using an example.

Caution: In this mode, it is mandatory to use the Common Data Register to read the regular data with a single read access. On the contrary, master-slave data coherency is not guaranteed.

Figure 147. Interleaved single channel CH0 with injected sequence CH11, CH12

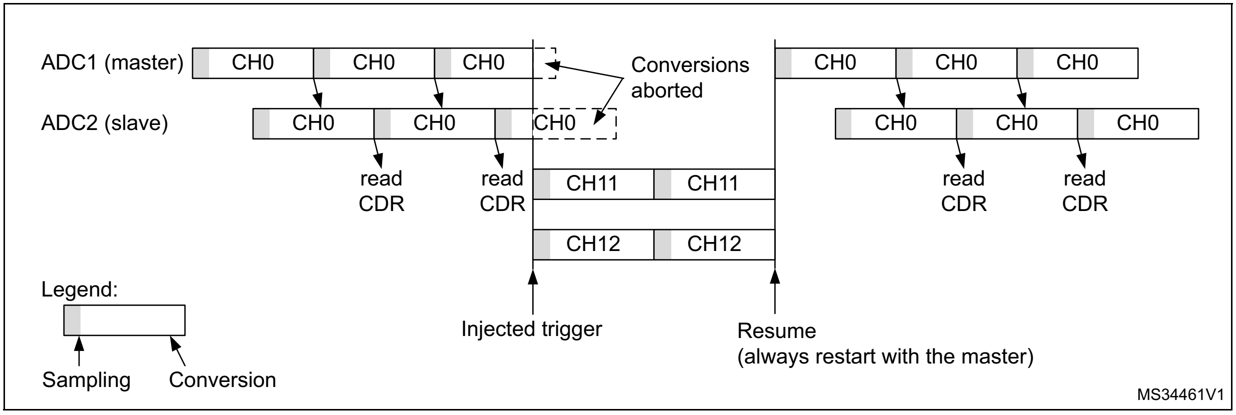


Figure 148. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first

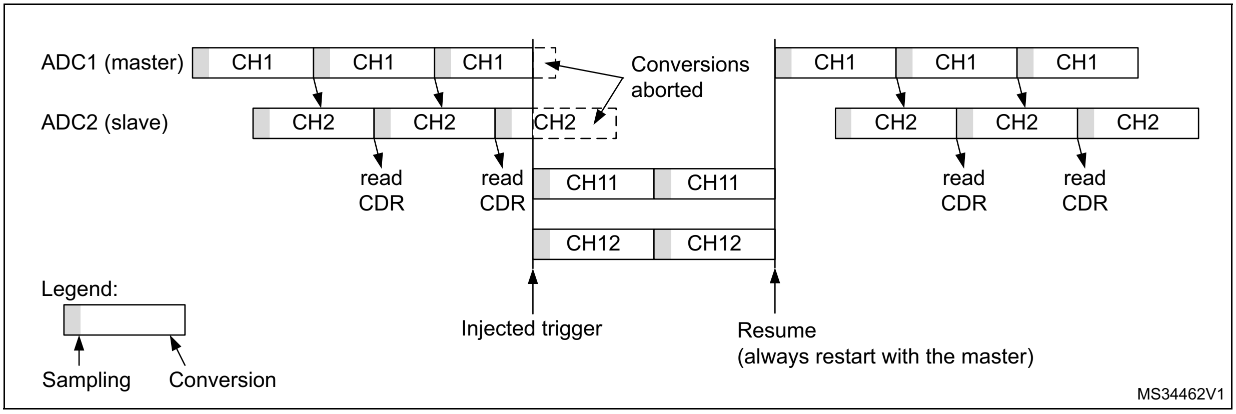
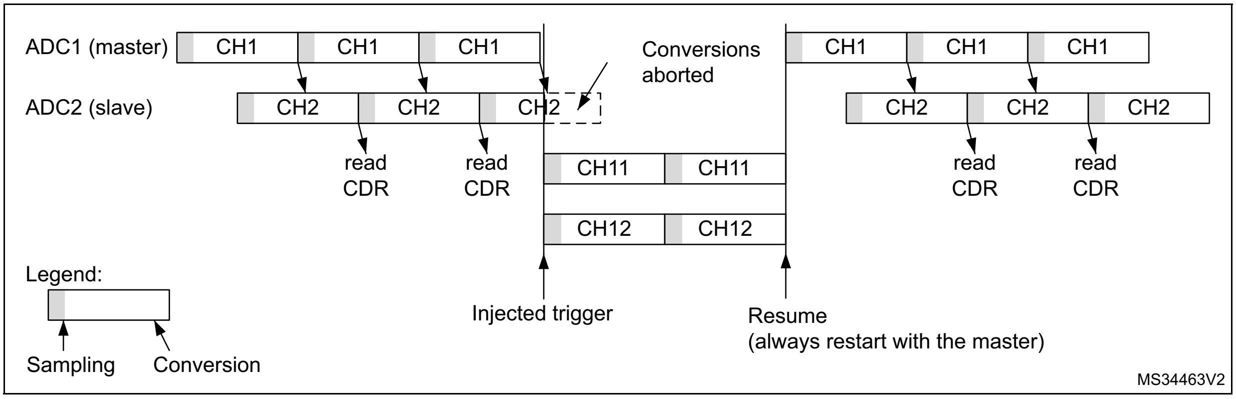


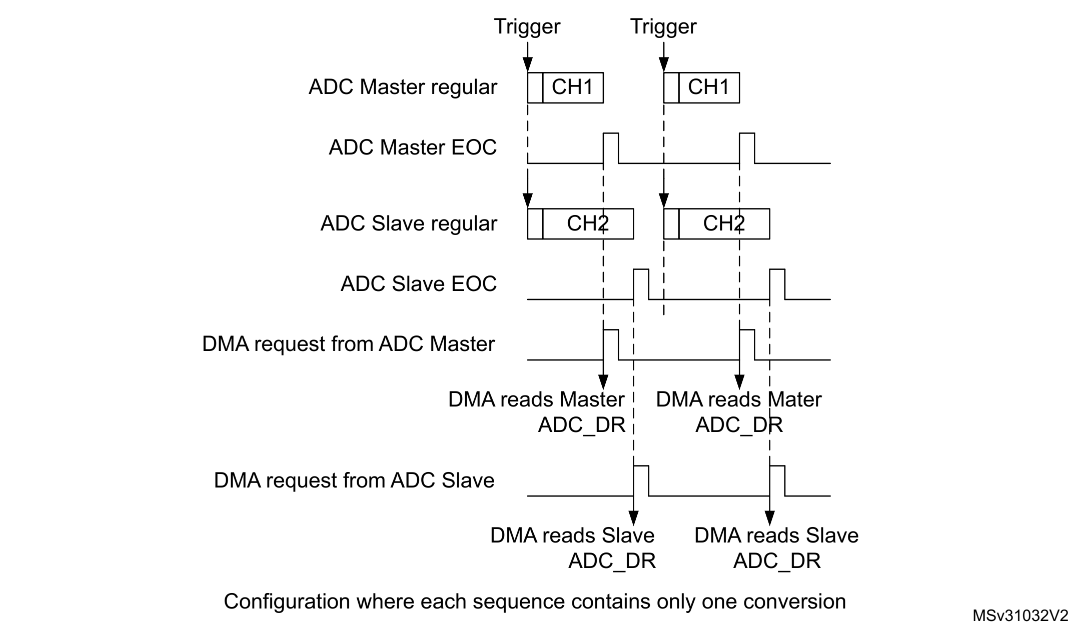
Figure 149. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 150: DMA Requests in regular simultaneous mode when MDMA=0b00*).

Figure 150. DMA Requests in regular simultaneous mode when MDMA=0b00



In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this MDMA bits must be configured in the ADCx_CCR register:

- **MDMA=0b10**: A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCx_CDR contains the two half-words representing two ADC-converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.
  This mode is used in interleaved mode and in regular simultaneous mode when resolution is 10-bit or 12-bit.

  Example:

  Interleaved dual mode: a DMA request is generated each time 2 data items are available:

  1st DMA request: ADCx_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

  2nd DMA request: ADCx_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 151. DMA requests in regular simultaneous mode when MDMA=0b10

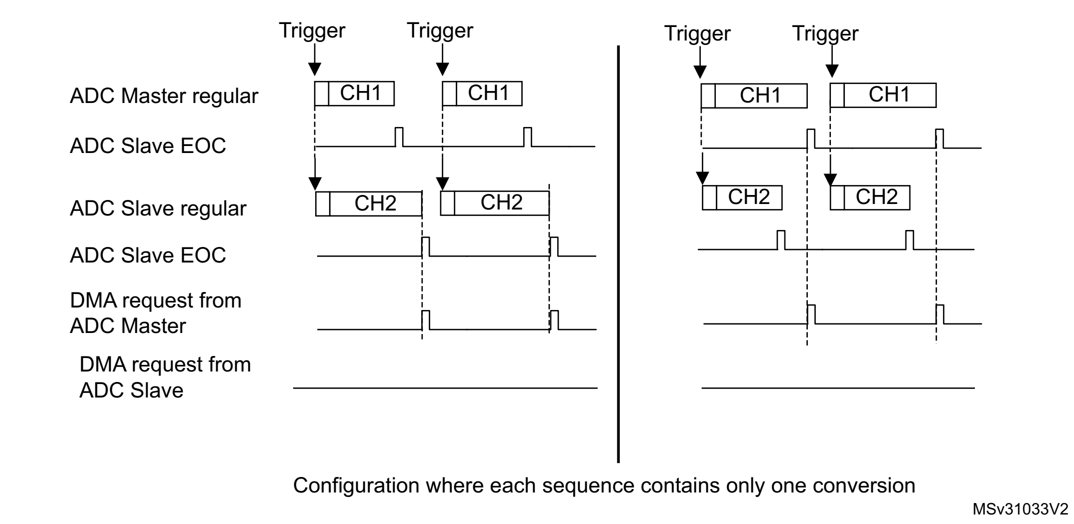
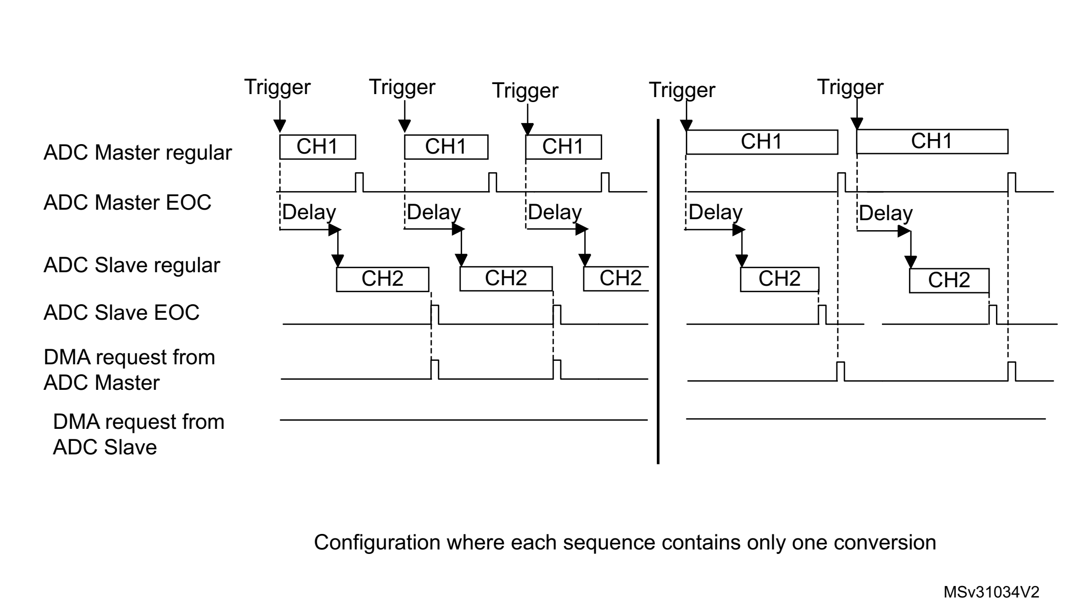

Trigger
Trigger
Trigger
Trigger
ADC Master regular
ADC Slave EOC
ADC Slave regular
ADC Slave EOC
DMA request from ADC Master
DMA request from ADC Slave
CH1
CH1
CH2
CH2
Configuration where each sequence contains only one conversion
MSv31033V2

Figure 152. DMA requests in interleaved mode when MDMA=0b10



Note: When using MDMA mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- **MDMA=0b11**: This mode is similar to the MDMA=0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.

  This mode is used in interleaved and regular simultaneous mode when resolution is 6-bit or when resolution is 8-bit and data is not signed (offsets must be disabled for all the involved channels).

  Example:

  Interleaved dual mode: a DMA request is generated each time 2 data items are available:

  1st DMA request: ADCx_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

  2nd DMA request: ADCx_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

Overrun detection

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the MDMA configuration). It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

DMA one shot mode/ DMA circular mode when MDMA mode is selected

When MDMA mode is selected (0b10 or 0b11), bit DMACFG of the ADCx_CCR register must also be configured to select between DMA one shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA* (bits DMACFG of master and slave ADC_CFGR are not relevant).

Stopping the conversions in dual ADC modes

The user must set the control bits ADSTP/JADSTP of the master ADC to stop the conversions of both ADC in dual ADC mode. The other ADSTP control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits ADSTART/JADSTART of the master and slave ADCs are both cleared by hardware.

21.4.31 Temperature sensor

The temperature sensor can be used to measure the junction temperature (Tj) of the device. The temperature sensor is internally connected to the ADC input channels which are used to convert the sensor output voltage to a digital value. When not in use, the sensor can be put in power down mode. It support the temperature range –40 to 125 °C.

Figure 153 shows the block diagram of connections between the temperature sensor and the ADC.

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.

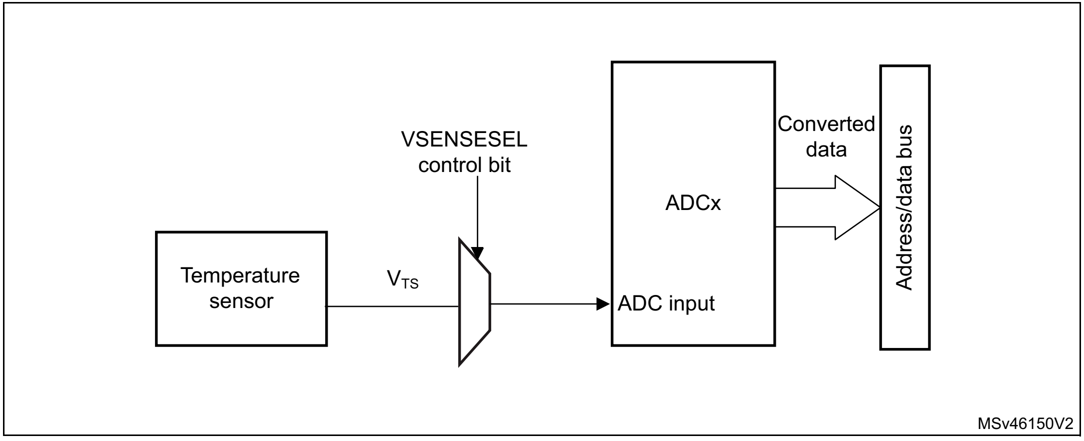
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel which is used to convert the sensor’s output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

Figure 153 shows the block diagram of the temperature sensor.

Figure 153. Temperature sensor channel block diagram



Reading the temperature

To use the sensor:

1. Select the ADC input channels that is connected to V_{TS}.
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the VSENSESEL bit in the ADCx_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{TS} data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in °C)} = \frac{\text{TS\_CAL2\_TEMP} - \text{TS\_CAL1\_TEMP}}{\text{TS\_CAL2} - \text{TS\_CAL1}} \times (\text{TS\_DATA} - \text{TS\_CAL1}) + 30\ \text{°C}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by ADC.

  Refer to the 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 VSENSESEL bits should be set at the same time.

The above formula is given for TS_DATA measurement done with the same V_{REF+}voltage as TS_CAL1/TS_CAL2 values. If V_{REF+} is different, the formula must be adapted. For example if V_{REF+} = 3.3 V and TS_CAL data are acquired at V_{REF+}= 3.0 V, TS_DATA must be replaced by TS_DATA x (3.3/3.0).

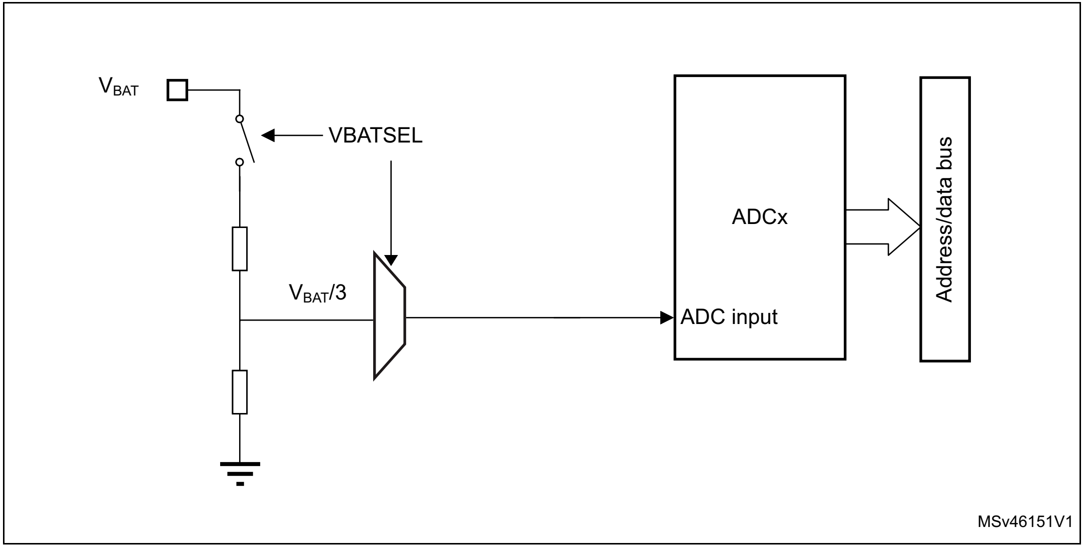
21.4.32 V_{BAT} supply monitoring

The VBATSEL bit in the ADCx_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage could be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 3. This bridge is automatically enabled when VBATSEL is set, to connect V_{BAT}/3 to the ADC input channels. As a consequence, the converted digital value is one third of the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the V_{BAT}/3 voltage.

The figure below shows the block diagram of the V_{BAT} sensing feature.

Figure 154. V_{BAT} channel block diagram



1. The VBATSEL bit must be set to enable the conversion of internal channel for V_{BAT}/3.

21.4.33 Monitoring the internal voltage reference

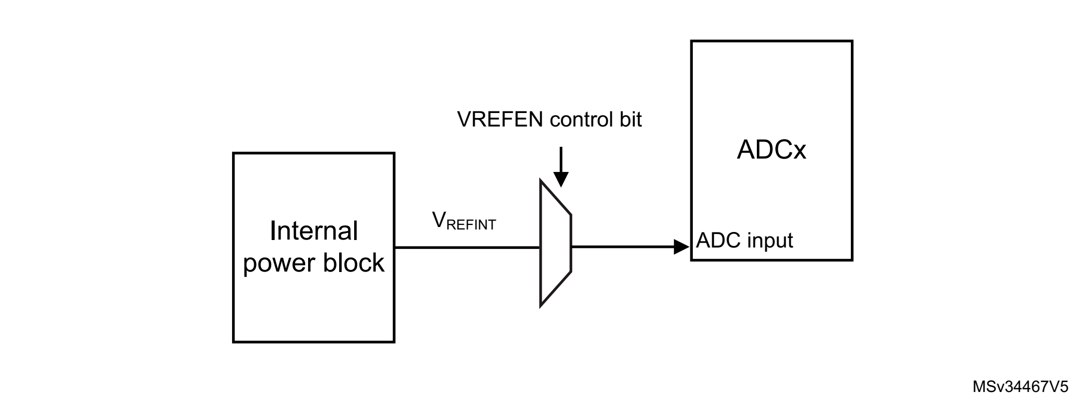
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

The internal reference voltage (V_{REFINT}) is internally connected to ADC1_INP18, ADC3_INP18, ADC4_INP18 and ADC5_INP18.

Refer to the electrical characteristics section of the product datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 155 shows the block diagram of the V_{REFINT} sensing feature.

Figure 155. V_{REFINT} channel block diagram



1. The VREFEN bit into ADCx_CCR register must be set to enable the conversion of internal channels (V_{REFINT}).

Calculating the actual V_{REF+} voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When V_{DDA} is connected to V_{REF+}, it is possible to compute the actual V_{DDA} voltage using the embedded internal reference voltage (V_{REFINT}). V_{REFINT} and its calibration data, acquired by the ADC during the manufacturing process at V_{DDA_Charac}, can be used to evaluate the actual V_{DDA} 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 V_{REFINT} calibration value
- VREFINT_DATA is the actual V_{REFINT} 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 V_{REF+} and the voltage applied on the converted channel.

For most applications V_{DDA} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{DDA}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL\_SCALE} \times ADC\_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+\_Charac} \times VREFINT\_CAL \times ADC\_DATA}{VREFINT\_DATA \times FULL\_SCALE}$$

For applications where V_{REF+} is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL\_SCALE} \times ADC\_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process.
- VREFINT_CAL is the V_{REFINT} calibration value
- ADC_DATA is the value measured by the ADC on channel x (right-aligned)
- VREFINT_DATA is the actual V_{REFINT} output value converted by the ADC
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, it will be 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* 16-*bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.*

21.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 174. ADC interrupts per each ADC

Interrupt event	Event flag	Enable control bit
ADC ready	ADRDY	ADRDYIE
End of conversion of a regular group	EOC	EOCIE
End of sequence of conversions of a regular group	EOS	EOSIE
End of conversion of a injected group	JEOC	JEOCIE
End of sequence of conversions of an injected group	JEOS	JEOSIE
Analog watchdog 1 status bit is set	AWD1	AWD1IE
Analog watchdog 2 status bit is set	AWD2	AWD2IE
Analog watchdog 3 status bit is set	AWD3	AWD3IE
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE
Injected context queue overflows	JQOVF	JQOVFIE

21.6 ADC registers (for each ADC)

Refer to *Section 1.2 on page 73* for a list of abbreviations used in register descriptions.

21.6.1 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.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	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	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to *Section 21.4.21: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software. writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed 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 regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

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 (bit 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

21.6.2 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.	Res.	Res.	Res.	JQOVF IE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMP IE	ADRDY IE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which 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 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which 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 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

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=0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular 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=0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular 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=0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

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=0 (which ensures that no regular 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=0 and JADSTART=0 (which ensures that no conversion is ongoing).

21.6.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	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.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.

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 in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN=0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

0: Writing ADCAL will launch a calibration in single-ended inputs mode.

1: Writing ADCAL will launch a calibration in differential inputs mode.

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bits is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled

1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 21.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set JADSTP only when JADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC)

In Auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set ADSTP only when ADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

In dual ADC regular simultaneous mode and interleaved mode, the bit ADSTP of the master ADC must be used to stop regular conversions. The other ADSTP bit is inactive.

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion will start immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (JEXTSEL=0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.

– in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN[1:0], a conversion will start immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (EXTSEL=0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.

– in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

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)

In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

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: The software is allowed to set ADDIS only when ADEN=1 and both ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC will be effectively ready to operate once the flag ADRDY 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, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)

21.6.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
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
ALIGN	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL4	EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	RES[1:0]		Res.	DMACFG	DMAEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **JQDIS**: Injected Queue disable

These bits are set and cleared by software to disable the Injected Queue mechanism :

0: Injected Queue enabled

1: Injected Queue disabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).

A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 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 AWD1 (available on ADC1 only)

00001: ADC analog input channel 1 monitored by AWD1

.....

10010: ADC analog input channel 18 monitored by AWD1

others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 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 AWD1CH[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 these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: JSQR mode 0: The Queue is never empty and maintains the last written configuration into JSQR.

1: JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 21.4.21: Queue of context for injected conversions* for more information.

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel
001: 2 channels
...
111: 8 channels

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled
1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to *Section : Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)*.

0: Right alignment
1: Left alignment

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off
1: Auto-delayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular 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 DISCEN=1 and CONT=1.

The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is 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 ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched 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 these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: Event 0

00001: Event 1

00010: Event 2

00011: Event 3

00100: Event 4

00101: Event 5

00110: Event 6

00111: Event 7

...

11111: Event 31

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit

01: 10-bit

10: 8-bit

11: 6-bit

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 2 Reserved, must be kept at reset value.

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 : Managing conversions using the DMA*

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bit DMACFG of the ADCx_CCR register.

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA to manage automatically the converted data. For more details, refer to *Section : Managing conversions using the DMA*.

0: DMA disabled

1: DMA enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bits MDMA[1:0] of the ADCx_CCR register.

21.6.5 ADC configuration register 2 (ADC_CFGR2)

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.	SMPTRIG	BULB	SWTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMP
				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.	ROVSM	TROVS	OVSS[3:0]				OVSR[2:0]			JOVSE	ROVSE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled

1: Sampling time control trigger mode enabled

The sampling time starts on the trigger rising edge, and the conversion on the trigger falling edge.

EXTEN[1:0] bits should be set to 01. BULB bit must not be set when the SMPTRIG bit is set.

When EXTEN[1:0] bits are set to 00, set SWTRIG to start the sampling and clear SWTRIG bit to start the conversion.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 26 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

SAMPTRIG bit must not be set when the BULB bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 25 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Software trigger starts the conversion for sampling time control trigger mode

1: Software trigger starts the sampling for sampling time control trigger mode

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **GCOMP**: Gain compensation mode

This bit is set and cleared by software to enable the gain compensation mode.

0: Regular ADC operating mode

1: Gain compensation enabled and applied on all channels

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular Oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 9 **TROVS**: Triggered Regular Oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

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

Other codes reserved

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 4:2 **OVSR[2:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the 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 these bits only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 1 **JOVSE**: Injected Oversampling Enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected Oversampling disabled

1: Injected Oversampling enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

Bit 0 **ROVSE**: Regular Oversampling Enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular Oversampling disabled

1: Regular Oversampling enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

21.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SMPPLUS	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
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
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SMPPLUS:** Addition of one clock cycle to the sampling time

1: 2.5 ADC clock cycle sampling time becomes 3.5 ADC clock cycles for the ADC_SMPR1 and ADC_SMPR2 registers.

0: The sampling time remains set to 2.5 ADC clock cycles remains

To make sure no conversion is ongoing, the software is allowed to write this bit only when ADSTART= 0 and JADSTART= 0.

Bit 30 Reserved, must be kept at reset value.

Bits 29:0 **SMP[9:0][2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 6.5 ADC clock cycles
010: 12.5 ADC clock cycles
011: 24.5 ADC clock cycles
100: 47.5 ADC clock cycles
101: 92.5 ADC clock cycles
110: 247.5 ADC clock cycles
111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

21.6.7 ADC sample time register 2 (ADC_SMPR2)

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.	SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					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
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:0 **SMP[18:10][2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 6.5 ADC clock cycles
010: 12.5 ADC clock cycles
011: 24.5 ADC clock cycles
100: 47.5 ADC clock cycles
101: 92.5 ADC clock cycles
110: 247.5 ADC clock cycles
111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

21.6.8 ADC watchdog threshold register 1 (ADC_TR1)

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.	AWDFILT[2:0]			LT1[11:0]											
	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 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1. Refer to *Section 21.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

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

Bits 14:12 **AWDFILT**: Analog watchdog filtering parameter

This bit is set and cleared by software.

000: No filtering

001: two consecutive detection generates an AWDx flag or an interrupt

...

111: Eight consecutive detection generates an AWDx flag or an interrupt

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

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 1.

Refer to *Section 21.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

21.6.9 ADC watchdog threshold register 2 (ADC_TR2)

Address offset: 0x24

Reset value: 0x00FF 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.	HT2[7:0]							
								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.	Res.	Res.	LT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT2[7:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to *Section 21.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT2[7:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to *Section 21.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

21.6.10 ADC watchdog threshold register 3 (ADC_TR3)

Address offset: 0x28

Reset value: 0x00FF 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.	HT3[7:0]							
								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.	Res.	Res.	LT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to *Section 21.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

21.6.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			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**
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 1st in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.12 ADC regular sequence register 2 (ADC_SQR2)

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.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			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
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 9th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 8th in the regular conversion sequence

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 7th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 6th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 5th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.13 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			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**
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 14th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 13th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 12th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 11th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 10th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.14 ADC regular sequence register 4 (ADC_SQR4)

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.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 16th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 15th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.15 ADC regular 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**
RDATA[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 **RDATA[15:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 21.4.26: Data management*.

21.6.16 ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
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**
JSQ20	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the injected conversion sequence.c

Note: The software is allowed to write these bits only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bits 8:7 **JEXTEN[1:0]**: External Trigger Enable and Polarity Selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS=0 (queue enabled), Hardware and software trigger detection disabled
00: If JQDIS=1 (queue disabled), Hardware trigger detection disabled (conversions can be launched 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 these bits only when JADSTART=0 (which ensures that no injected conversion is ongoing).

If JQM=1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 21.4.21: Queue of context for injected conversions)

Bits 6:2 **JEXTSEL[4:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: Event 0
00001: Event 1
00010: Event 2
00011: Event 3
00100: Event 4
00101: Event 5
00110: Event 6
00111: Event 7
...
11111: Event 31

Note: The software is allowed to write these bits only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JL[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion
01: 2 conversions
10: 3 conversions
11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.17 ADC offset y register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y -1), (y= 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSETy _EN	OFFSETy_CH[4:0]					SATEN	OFFSE TPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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**
Res.	Res.	Res.	Res.	OFFSETy[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OFFSETy_EN:** Offset y enable

This bit is written by software to enable or disable the offset programmed into bits OFFSETy[11:0].

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 30:26 **OFFSETy_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[11:0] will apply.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the data offset y.

Bit 25 **SATEN**: Saturation enable

This bit is set and cleared by software to enable the saturation at 0x000 and 0xFFF for the offset function.

0: No saturation control, offset result can be signed

1: Saturation enabled, offset result unsigned and saturated at 0x000 and 0xFFF

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 24 **OFFSETPOS**: Positive offset

This bit is set and cleared by software to enable the positive offset.

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSETy[11:0]:** Data offset y for the channel programmed into bits OFFSETy_CH[4:0]

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.

Ex: if OFFSET1_CH[4:0]=4 and OFFSET2_CH[4:0]=4, this is OFFSET1[11:0] which is subtracted when converting channel 4.

21.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y - 1), (y = 1 to 4)

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
JDATA[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 **JDATA[15:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in *Section 21.4.26: Data management*.

21.6.19 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.	AWD2CH[18:16]		
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **AWD2CH[18:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWD2CH[18:0] = 000..0, the analog watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the analog watchdog.

21.6.20 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.	AWD3CH[18:16]		
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **AWD3CH[18:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel i is not monitored by AWD3
AWD3CH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWD3CH[18:0] = 000..0, the analog watchdog 3 is disabled

Note: The channels selected by AWD3CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the analog watchdog.

21.6.21 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0xB0

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.	DIFSEL[18:16]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIFSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **DIFSEL[18:0]**: Differential mode for channels 18 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or differential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single ended mode
DIFSEL[i] = 1: ADC analog input channel i is configured in differential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).

The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

21.6.22 ADC calibration factors (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.	CALFACT_D[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.	CALFACT_S[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **CALFACT_D[6:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs 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 will then be applied once a new differential calibration is launched.

Note: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

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

Bits 6:0 **CALFACT_S[6:0]**: Calibration Factors In single-ended mode

These bits are written by hardware or by software.

Once a single-ended inputs 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 will then be applied once a new single-ended calibration is launched.

Note: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

21.6.23 ADC Gain compensation Register (ADC_GCOMP)

Address offset: 0xC0

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.	GCOMPCOEFF[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **GCOMPCOEFF[13:0]**: Gain compensation coefficient

These bits are set and cleared by software to program the gain compensation coefficient.

00 1000 0000 0000: gain factor of 0.5

...

01 0000 0000 0000: gain factor of 1

10 0000 0000 0000: gain factor of 2

11 0000 0000 0000: gain factor of 3

...

The coefficient is divided by 4096 to get the gain factor ranging from 0 to 3.999756.

Note: This gain compensation is only applied when GCOMP bit of ADC_CFGR2 register is 1.

21.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

21.7.1 ADCx common status register (ADCx_CSR) (x=12 or 345)

Address offset: 0x00 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR register.

One interface controls ADC1 and ADC2, while the other interface controls ADC3, ADC4 and ADC5.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
					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.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **JQOVF_SLV:** Injected Context Queue Overflow flag of the slave ADC

This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC

This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC

This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC

This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17 **EOSMP_SLV:** End of Sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADC_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF_MST:** Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

21.7.2 ADCx common control register (ADCx_CCR) (x=12 or 345)

Address offset: 0x08 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

One interface controls ADC1 and ADC2, while the other interface controls ADC3, ADC4 and ADC5.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATSEL	VSENSESEL	VREFEN	PRESC[3:0]				CKMODE[1:0]	
							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
MDMA[1:0]		DMACFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATSEL**: VBAT selection

This bit is set and cleared by software to control VBAT.

0: V_{BAT} channel disabled.
1: V_{BAT} channel enabled

Bit 23 **VSENSESEL**: V_{TS} selection

This bit is set and cleared by software to control V_{TS}.

0: Temperature sensor channel disabled
1: Temperature sensor channel enabled

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} channel.

0: V_{REFINT} channel disabled
1: V_{REFINT} channel enabled

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

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: The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0). The ADC prescaler value is applied only when CKMODE[1:0] = 0b00.

Bits 17:16 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: adc_ker_ck (x=123) (Asynchronous clock mode), generated at product level (refer to *Section 6: Reset and clock control (RCC)*)

01: adc_hclk/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set to 1 (HPRE[3:0] = 0xxx in RCC_CFGR register) and if the system clock has a 50% duty cycle.

10: adc_hclk/2 (Synchronous clock mode)

11: adc_hclk/4 (Synchronous clock mode)

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 ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 15:14 **MDMA[1:0]:** Direct memory access mode for dual ADC mode

This bitfield is set and cleared by software. Refer to the DMA controller section for more details.

00: MDMA mode disabled

01: Reserved

10: MDMA mode enabled for 12 and 10-bit resolution

11: MDMA mode enabled for 8 and 6-bit resolution

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 13 **DMACFG**: DMA configuration (for dual ADC mode)

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 : Managing conversions using the DMA*

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY:** Delay between 2 sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 175* for the value of ADC resolution versus DELAY bits values.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode.
All the ADCs independent:
00000: Independent mode

00001 to 01001: Dual mode, master and slave ADCs working together
00001: Combined regular simultaneous + injected simultaneous mode
00010: Combined regular simultaneous + alternate trigger mode
00011: Combined Interleaved mode + injected simultaneous mode
00100: Reserved
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
All other combinations are reserved and must not be programmed

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Table 175. DELAY bits versus ADC resolution

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * $T_{adc\_ker\_ck}$	1 * $T_{adc\_ker\_ck}$	1 * $T_{adc\_ker\_ck}$	1 * $T_{adc\_ker\_ck}$
0001	2 * $T_{adc\_ker\_ck}$	2 * $T_{adc\_ker\_ck}$	2 * $T_{adc\_ker\_ck}$	2 * $T_{adc\_ker\_ck}$
0010	3 * $T_{adc\_ker\_ck}$	3 * $T_{adc\_ker\_ck}$	3 * $T_{adc\_ker\_ck}$	3 * $T_{adc\_ker\_ck}$
0011	4 * $T_{adc\_ker\_ck}$	4 * $T_{adc\_ker\_ck}$	4 * $T_{adc\_ker\_ck}$	4 * $T_{adc\_ker\_ck}$
0100	5 * $T_{adc\_ker\_ck}$	5 * $T_{adc\_ker\_ck}$	5 * $T_{adc\_ker\_ck}$	5 * $T_{adc\_ker\_ck}$
0101	6 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$
0110	7 * $T_{adc\_ker\_ck}$	7 * $T_{adc\_ker\_ck}$	7 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$
0111	8 * $T_{adc\_ker\_ck}$	8 * $T_{adc\_ker\_ck}$	8 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$
1000	9 * $T_{adc\_ker\_ck}$	9 * $T_{adc\_ker\_ck}$	8 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$
1001	10 * $T_{adc\_ker\_ck}$	10 * $T_{adc\_ker\_ck}$	8 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$
1010	11 * $T_{adc\_ker\_ck}$	10 * $T_{adc\_ker\_ck}$	8 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$
1011	12 * $T_{adc\_ker\_ck}$	10 * $T_{adc\_ker\_ck}$	8 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$
others	12 * $T_{adc\_ker\_ck}$	10 * $T_{adc\_ker\_ck}$	8 * $T_{adc\_ker\_ck}$	6 * $T_{adc\_ker\_ck}$

21.7.3 ADCx common regular data register for dual mode (ADCx_CDR) (x=12 or 345)

Address offset: 0x0C (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

One interface controls ADC1 and ADC2, while the other interface controls ADC3, ADC4 and ADC5.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_SLV[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
RDATA_MST[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 21.4.30: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 21.4.30: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)*)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

21.8 ADC register map

The following table summarizes the ADC registers.

Table 176. ADC global register map

Offset	Register
0x000 - 0x0FC	Master ADC1/ADC3
0x100 - 0x1FC	Slave ADC2/ADC4
0x200 - 0x2FC	Reserved/single ADC5
0x300 - 0x30C	Master and slave ADCs common registers

Table 177. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC)

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.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x04	**ADC_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x08	**ADC_CR**	ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1	0																							0	0	0	0	0	0
0x0C	**ADC_CFGR**	JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM [2:0]			DISCEN	ALIGN	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL4	EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	RES [1:0]		Res.	DMACFG	DMAEN
	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	0	0	0	0		0	0
0x10	**ADC_CFGR2**	Res.	Res.	Res.	Res.	SMPTRIG	BULB	SWTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMP	Res.	Res.	Res.	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				OVSR [2:0]			JOVSE	ROVSE
	Reset value					0	0	0									0						0	0	0	0	0	0	0	0	0	0	0
0x14	**ADC_SMPR1**	SMPPLUS	Res.	SMP9 [2:0]			SMP8 [2:0]			SMP7 [2:0]			SMP6 [2:0]			SMP5 [2:0]			SMP4 [2:0]			SMP3 [2:0]			SMP2 [2:0]			SMP1 [2:0]			SMP0 [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	0
0x18	**ADC_SMPR2**	Res.	Res.	Res.	Res.	Res.	SMP18 [2:0]			SMP17 [2:0]			SMP16 [2:0]			SMP15 [2:0]			SMP14 [2:0]			SMP13 [2:0]			SMP12 [2:0]			SMP11 [2:0]			SMP10 [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
0x1C	Reserved	Res.																															
0x20	**ADC_TR1**	Res.	Res.	Res.	Res.	HT1[11:0]												Res.	AWDFILT [2:0]			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_TR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x28	**ADC_TR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x2C	Reserved	Res.																															
0x30	**ADC_SQR1**	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	L[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
0x34	**ADC_SQR2**	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	SQ5[4:0]				
	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
0x38	**ADC_SQR3**	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	SQ10[4:0]				
	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
0x3C	**ADC_SQR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x40	**ADC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	regular RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 177. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (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
0x44- 0x48	Reserved	Res.																															
0x4C	ADC_JSQR	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]					Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL [4:0]					JL[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	0	0	0	0	0
0x50- 0x5C	Reserved	Res.																															
0x60	ADC_OFR1	OFFSET1_EN	OFFSET1_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET1[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x64	ADC_OFR2	OFFSET2_EN	OFFSET2_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET2[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x68	ADC_OFR3	OFFSET3_EN	OFFSET3_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET3[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x6C	ADC_OFR4	OFFSET4_EN	OFFSET4_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET4[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x70- 0x7C	Reserved	Res.																															
0x80	ADC_JDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	ADC_JDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	ADC_JDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	ADC_JDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C- 0x9C	Reserved	Res.																															
0xA0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[18:0]																		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[18:0]																		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8- 0xAC	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.

Table 177. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (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
0xB0	ADC_DIFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[18:0]																		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB4	ADC_CALFACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]						
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0xC0	ADC_GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMP[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 178. ADC register map and reset values (master and slave ADC common registers) offset = 0x300

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	ADCx_CSR	Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
							slave ADC2																master ADC1										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x04	Reserved	Res.																															
0x08	ADCx_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATSEL	VSENSESEL	VREFEN	PRESC[3:0]				CKMODE[1:0]		MDMA[1:0]		DMACFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0				0	0	0	0	0
0x0C	ADCx_CDR	RDATA_SLV[15:0]																RDATA_MST[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

Refer to *Section 2.2 on page 81* for the register boundary addresses.

22 Digital-to-analog converter (DAC)

22.1 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 up to two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. 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_OUTy 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 allow 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.

22.2 DAC main features

The DAC main features are the following (see *Figure 156: Dual-channel DAC block diagram*)

- Up to four DAC interfaces, maximum two output channels each
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Sawtooth wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DACx_OUTy output pin
- DAC output connection to on chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference, V_{REF+}

Figure 156 shows the block diagram of a DAC channel and *Table 180* gives the pin description.

22.3 DAC implementation

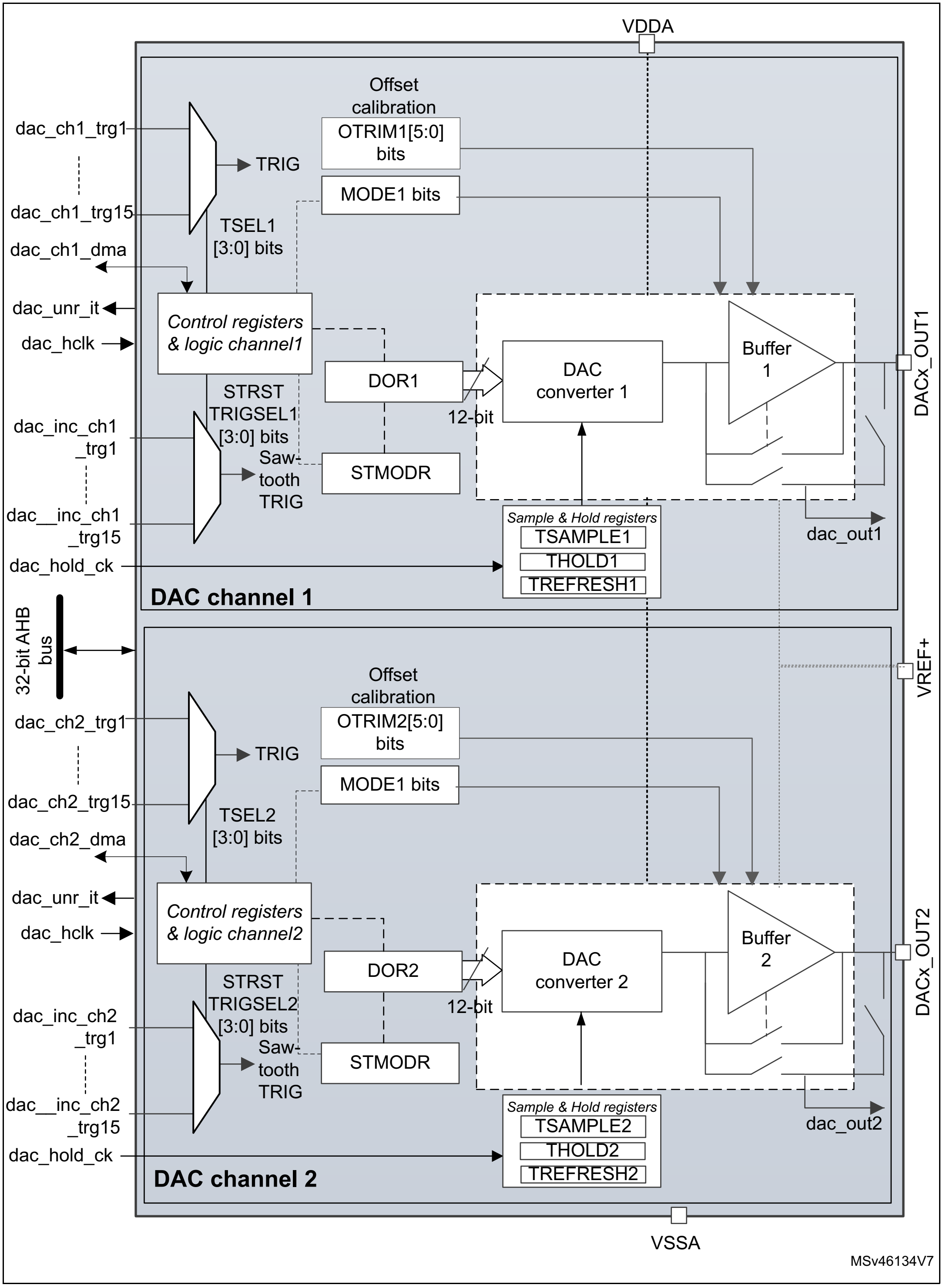
Table 179. DAC features

DAC features	DAC1	DAC2	DAC3	DAC4
Dual channel	X	-	X	X
Output buffer	X	X	-	-
I/O connection	DAC1_OUT1 on PA4 DAC1_OUT2 on PA5	DAC2_OUT1 on PA6	No connection to a GPIO	
Maximum sampling time	1MSPS		15MSPS	
Autonomous mode	-			

22.4 DAC functional description

22.4.1 DAC block diagram

Figure 156. Dual-channel DAC block diagram



1. MODEx bits in the DAC_MCR control the output mode and allow switching between the Normal mode in buffer/unbuffered configuration and the Sample and hold mode.
2. Refer to *Section 22.3: DAC implementation* for channel2 availability.
3. DAC channel2 is available only on DAC1, DAC3 and DAC4.

22.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DACx_OUTy can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx 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/LSE 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_OUTy 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 or LSE clock source (dac_hold_ck).

Table 180. 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)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DACx_OUTy	Analog output signal	DACx channely analog output

Table 181. DAC input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 15)	Inputs	DAC channel2 trigger inputs
dac_ch1_inc_trgx (x = 1 to 15)	Inputs	DAC channel1 sawtooth increment trigger inputs
dac_chn2_inc_trgx (x = 1 to 15)	Inputs	DAC channel1 sawtooth increment trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_hclk	Input	DAC peripheral clock
dac_hold_ck	Input	DAC low-power clock used in Sample and hold mode

Table 181. DAC input/output signals (continued)

Internal signal name	Signal type	Description
dac_out1	Analog output	DAC channel1 output for on-chip peripherals
dac_out2	Analog output	DAC channel2 output for on-chip peripherals

Table 182. DAC1 interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi or ck_lse (selected in the RCC)	LSI or LSE clock selected in the RCC
dac_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	EXTI9	External pin
dac_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_chx_trg9 (x = 1, 2)	hrtim_dac_reset_trg1	Internal signal from on-chip timers
dac_chx_trg10 (x = 1, 2)	hrtim_dac_reset_trg2	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	hrtim_dac_reset_trg3	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	hrtim_dac_reset_trg4	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	hrtim_dac_reset_trg5	Internal signal from on-chip timers
dac_chx_trg14 (x = 1, 2)	hrtim_dac_reset_trg6	Internal signal from on-chip timers
dac_chx_trg15 (x = 1, 2)	hrtim_dac_trg1	Internal signal from on-chip timers
dac_inc_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg6 (x = 1, 2)	EXTI10	External pin
dac_inc_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg9 (x = 1, 2)	hrtim_dac_step_trg1	Internal signal from on-chip timers
dac_inc_chx_trg10 (x = 1, 2)	hrtim_dac_step_trg2	Internal signal from on-chip timers
dac_inc_chx_trg11 (x = 1, 2)	hrtim_dac_step_trg3	Internal signal from on-chip timers
dac_inc_chx_trg12 (x = 1, 2)	hrtim_dac_step_trg4	Internal signal from on-chip timers

Table 182. DAC1 interconnection (continued)

Signal name	Source	Source type
dac_inc_chx_trg13 (x = 1, 2)	hrtim_dac_step_trg5	Internal signal from on-chip timers
dac_inc_chx_trg14 (x = 1, 2)	hrtim_dac_step_trg6	Internal signal from on-chip timers

Table 183. DAC2 interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi or ck_lse (selected in the RCC)	LSI or LSE clock selected in the RCC
dac_ch1_trg1	TIM8_TRGO	Internal signal from on-chip timers
dac_ch1_trg2	TIM7_TRGO	Internal signal from on-chip timers
dac_ch1_trg3	TIM15_TRGO	Internal signal from on-chip timers
dac_ch1_trg4	TIM2_TRGO	Internal signal from on-chip timers
dac_ch1_trg5	TIM4_TRGO	Internal signal from on-chip timers
dac_ch1_trg6	EXTI9	External pin
dac_ch1_trg7	TIM6_TRGO	Internal signal from on-chip timers
dac_ch1_trg8	TIM3_TRGO	Internal signal from on-chip timers
dac_ch1_trg9	hrtim_dac_reset_trg1	Internal signal from on-chip timers
dac_ch1_trg10	hrtim_dac_reset_trg2	Internal signal from on-chip timers
dac_ch1_trg11	hrtim_dac_reset_trg3	Internal signal from on-chip timers
dac_ch1_trg12	hrtim_dac_reset_trg4	Internal signal from on-chip timers
dac_ch1_trg13	hrtim_dac_reset_trg5	Internal signal from on-chip timers
dac_ch1_trg14	hrtim_dac_reset_trg6	Internal signal from on-chip timers
dac_ch1_trg15	hrtim_dac_trg2	Internal signal from on-chip timers
dac_inc_ch1_trg1	TIM8_TRGO	Internal signal from on-chip timers
dac_inc_ch1_trg2	TIM7_TRGO	Internal signal from on-chip timers
dac_inc_ch1_trg3	TIM15_TRGO	Internal signal from on-chip timers
dac_inc_ch1_trg4	TIM2_TRGO	Internal signal from on-chip timers
dac_inc_ch1_trg5	TIM4_TRGO	Internal signal from on-chip timers
dac_inc_ch1_trg6	EXTI10	External pin
dac_inc_ch1_trg7	TIM6_TRGO	Internal signal from on-chip timers
dac_inc_ch1_trg8	TIM3_TRGO	Internal signal from on-chip timers
dac_inc_ch1_trg9	hrtim_dac_step_trg1	Internal signal from on-chip timers
dac_inc_ch1_trg10	hrtim_dac_step_trg2	Internal signal from on-chip timers
dac_inc_ch1_trg11	hrtim_dac_step_trg3	Internal signal from on-chip timers
dac_inc_ch1_trg12	hrtim_dac_step_trg4	Internal signal from on-chip timers

Table 183. DAC2 interconnection (continued)

Signal name	Source	Source type
dac_inc_ch1_trg13	hrtim_dac_step_trg5	Internal signal from on-chip timers
dac_inc_ch1_trg14	hrtim_dac_step_trg6	Internal signal from on-chip timers

Table 184. DAC3 interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi or ck_lse (selected in the RCC)	LSI or LSE clock selected in the RCC
dac_chx_trg1 (x = 1, 2)	TIM1_TRGO	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	EXTI9	External pin
dac_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_chx_trg9 (x = 1, 2)	hrtim_dac_reset_trg1	Internal signal from on-chip timers
dac_chx_trg10 (x = 1, 2)	hrtim_dac_reset_trg2	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	hrtim_dac_reset_trg3	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	hrtim_dac_reset_trg4	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	hrtim_dac_reset_trg5	Internal signal from on-chip timers
dac_chx_trg14 (x = 1, 2)	hrtim_dac_reset_trg6	Internal signal from on-chip timers
dac_chx_trg15 (x = 1, 2)	hrtim_dac_trg3	Internal signal from on-chip timers
dac_inc_chx_trg1 (x = 1, 2)	TIM1_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers

Table 184. DAC3 interconnection (continued)

Signal name	Source	Source type
dac_inc_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg6 (x = 1, 2)	EXTI10	External pin
dac_inc_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg9 (x = 1, 2)	hrtim_dac_step_trg1	Internal signal from on-chip timers
dac_inc_chx_trg10 (x = 1, 2)	hrtim_dac_step_trg2	Internal signal from on-chip timers
dac_inc_chx_trg11 (x = 1, 2)	hrtim_dac_step_trg3	Internal signal from on-chip timers
dac_inc_chx_trg12 (x = 1, 2)	hrtim_dac_step_trg4	Internal signal from on-chip timers
dac_inc_chx_trg13 (x = 1, 2)	hrtim_dac_step_trg5	Internal signal from on-chip timers
dac_inc_chx_trg14 (x = 1, 2)	hrtim_dac_step_trg6	Internal signal from on-chip timers

Table 185. DAC4 interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi or ck_lse (selected in RCC)	LSI or LSE clock selected in the RCC
dac_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	EXTI9	External pin
dac_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers

Table 185. DAC4 interconnection (continued)

Signal name	Source	Source type
dac_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_chx_trg9 (x = 1, 2)	hrtim_dac_reset_trg1	Internal signal from on-chip timers
dac_chx_trg10 (x = 1, 2)	hrtim_dac_reset_trg2	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	hrtim_dac_reset_trg3	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	hrtim_dac_reset_trg4	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	hrtim_dac_reset_trg5	Internal signal from on-chip timers
dac_chx_trg14 (x = 1, 2)	hrtim_dac_reset_trg6	Internal signal from on-chip timers
dac_chx_trg15 (x = 1, 2)	hrtim_dac_trg1	Internal signal from on-chip timers
dac_inc_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg6 (x = 1, 2)	EXTI10	External pin
dac_inc_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg9 (x = 1, 2)	hrtim_dac_step_trg1	Internal signal from on-chip timers
dac_inc_chx_trg10 (x = 1, 2)	hrtim_dac_step_trg2	Internal signal from on-chip timers
dac_inc_chx_trg11 (x = 1, 2)	hrtim_dac_step_trg3	Internal signal from on-chip timers
dac_inc_chx_trg12 (x = 1, 2)	hrtim_dac_step_trg4	Internal signal from on-chip timers

Table 185. DAC4 interconnection (continued)

Signal name	Source	Source type
dac_inc_chx_trg13 (x = 1, 2)	hrtim_dac_step_trg5	Internal signal from on-chip timers
dac_inc_chx_trg14 (x = 1, 2)	hrtim_dac_step_trg6	Internal signal from on-chip timers

22.4.3 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

22.4.4 DAC data format

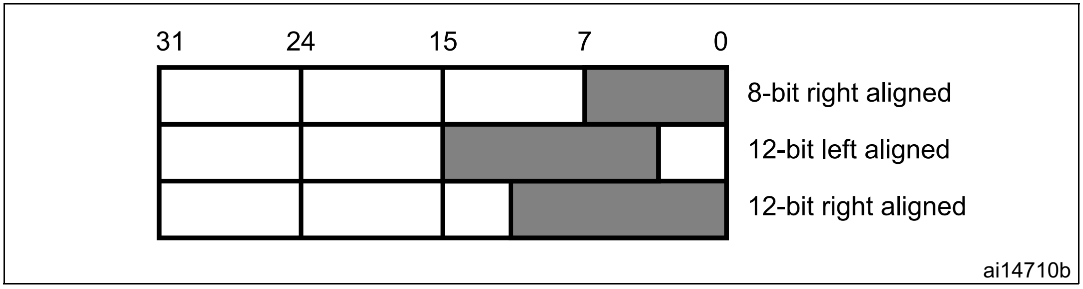
Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:
  - 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)
  - 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
  - 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

Figure 157. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

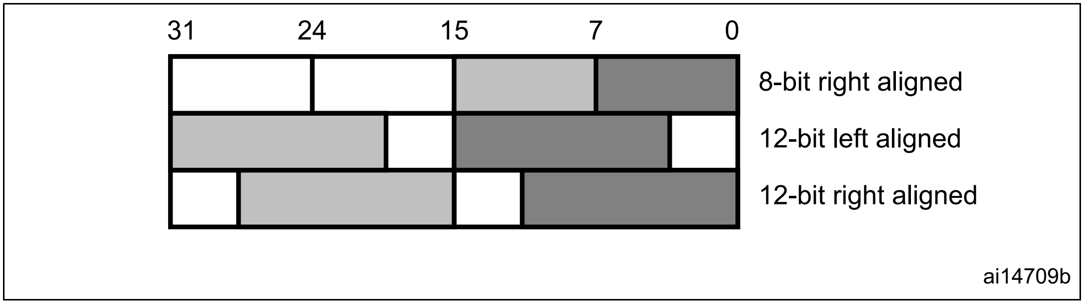
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

  – 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 158. Data registers in dual DAC channel mode



Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2’s complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB bit of the data written to DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB bit is inverted. The four LSBs are simply ignored.

Table 186. Data format (case of 12-bit data)

SINFORMATx bit	DATA written to DHRx register	DATA transfered to DORx register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF

Table 186. Data format (case of 12-bit data) (continued)

SINFORMATx bit	DATA written to DHRx register	DATA transfered to DORx register
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

22.4.5 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_hclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_hclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

HFSEL bits of DAC_MCR must be set when dac_hclk clock speed is faster than 80 MHz. It adds an extra delay to the transfer from DAC_DHRx register to DAC_DORx register.

Refer to Table *HFSEL description* below for the limitation of the DAC_DORx update rate depending on HFSEL bits and dac_hclk clock frequency.

If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

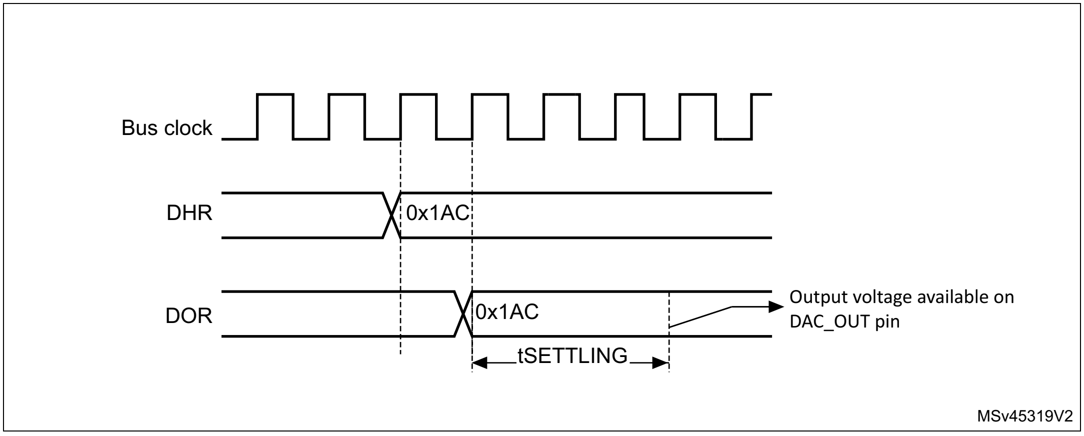
The above timing is only related to the limitation of the DAC interface. Refer also to the $t_{SETTLING}$ parameter value in the product datasheet.

Table 187. HFSEL description

HFSEL[1:0]	AHB frequency	Function
00	≤ 80 MHz	DAC_DOR update rate up to 3 AHB clock cycles
01	>80 MHz[1]	DAC_DOR update rate up to 5 AHB clock cycles
10	>160 MHz	DAC_DOR update rate up to 7 AHB clock cycles
11	Reserved	-

1. Refer to the device datasheet for the value of the maximum AHB frequency.

Figure 159. Timing diagram for conversion with trigger disabled TEN = 0



22.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACoutput} = V_{REF} \times \frac{\text{DOR}}{4096}$$

22.4.7 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[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 22.4.2: 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_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_hclk 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_DORx register has been loaded with the DAC_DHRx register contents.

The reset trigger selection and the increment trigger selection of the sawtooth generation are performed through STRSTTRIGSELx and STINCTRIGSELx control bits, respectively. STRSTTRIGSELx mapping is similar to TSELx. Refer to the *Section 22.4.2: DAC pins and internal signals* for TSELx, STRSTTRIGSELx, and STINCTRIGSELx mappings.

Note: *TSELx[3:0] bit cannot be changed when the ENx bit is set.*

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_hclk clock cycle.

22.4.8 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx 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 channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx 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 each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

DMA Double data mode

When the DMA controller is used in Normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.
2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA Double data mode.

In DMA Double mode, DMA requests can only handle one DAC channel. To use two channel outputs in DMA Double mode, each DMA channel has to be configured separately.

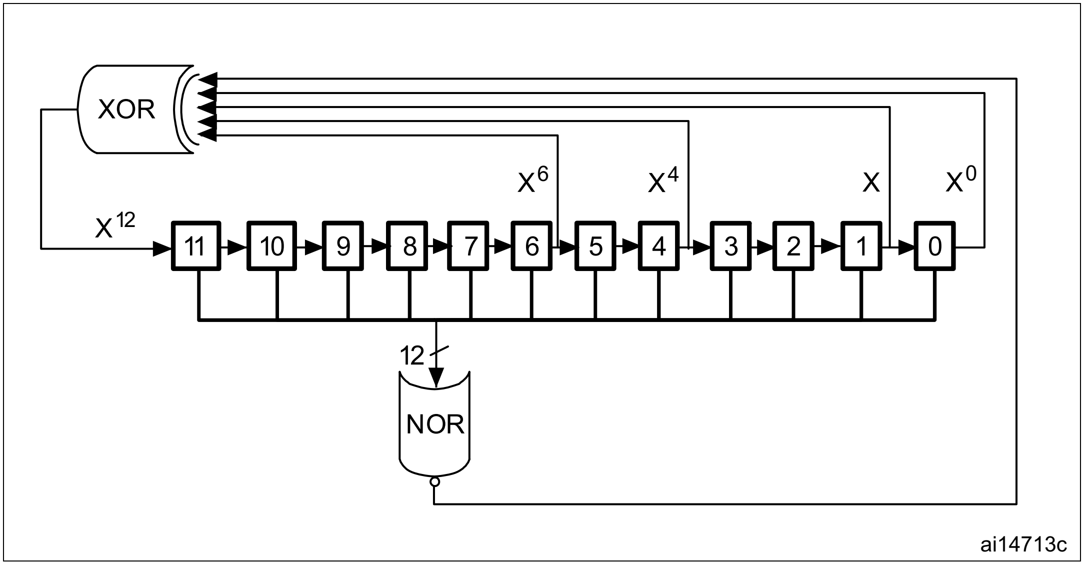
The following conditions must be met to change from Double data to single data mode or vice versa:

- The DAC must be disabled.
- DMAEN bit must be cleared (ENx = 0 and DMAEN = 0).

22.4.9 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 WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_hclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 160. DAC LFSR register calculation algorithm

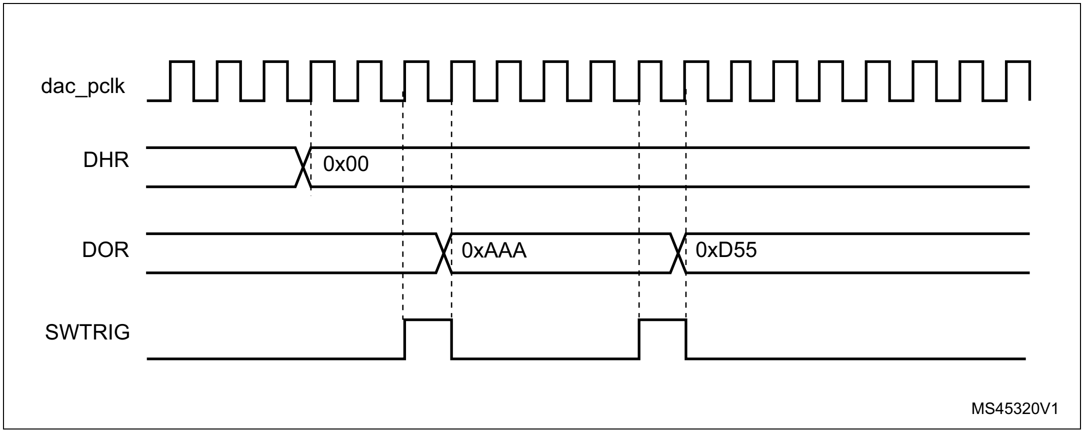


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a ‘1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 161. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

22.4.10 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 WAVEx[1:0] to 10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_hclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[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 WAVEx[1:0] bits.

Figure 162. DAC triangle wave generation

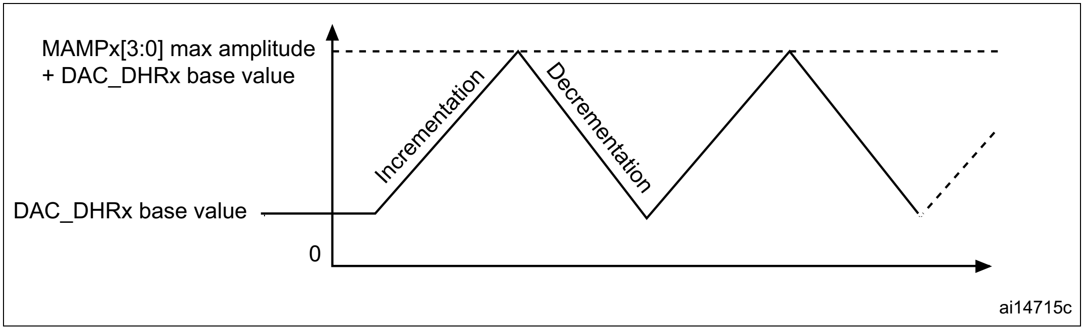
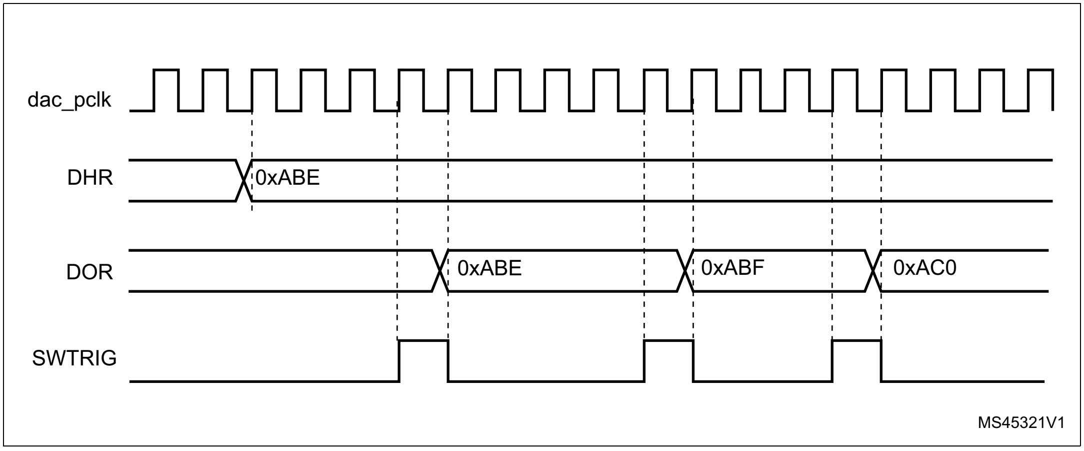


Figure 163. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

22.4.11 DAC sawtooth wave generation

The DAC can generate a sawtooth waveform. Specific register settings for the initial value, increment value and direction control are required:

- DAC sawtooth wave generation is selected by setting WAVEx[1:0] to 11 in the DAC_CR register.
- The sawtooth counter initial value (reset value) is configured through STRSTDATAx[11:0] bits in the DAC_STRx register.
- The increment value is defined by the STINCDATAx[15:0] bits in the DAC_STRx register.
- The sawtooth direction is defined by STDIRx bit in the DAC_STRx register.

The sawtooth counter starts from STRSTDATAx[11:0] (bits 12 to 15 are set to 0000), each increment trigger then increments (or decrements) STINCDATAx[15:0] value.

The DAC output is used from 12 MSB of those counter value. When the counter reaches 0x0000 or 0xFFFF, the value is saturated. The sawtooth reset trigger signal initializes the counter value to the STRSTDATAx[11:0] (bits 12 to 15 are set to 0000) value.

The increment trigger and reset trigger must be selected through the STINCTRIGSELx[3:0] and the STRSTTRIGSELx[3:0] bits.

Figure 164. DAC sawtooth wave generation (STDIRx = 0)

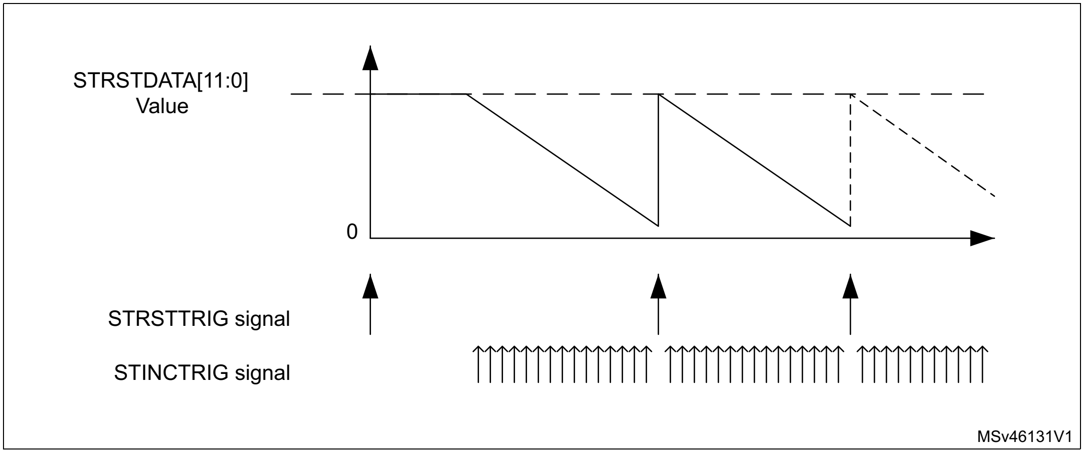
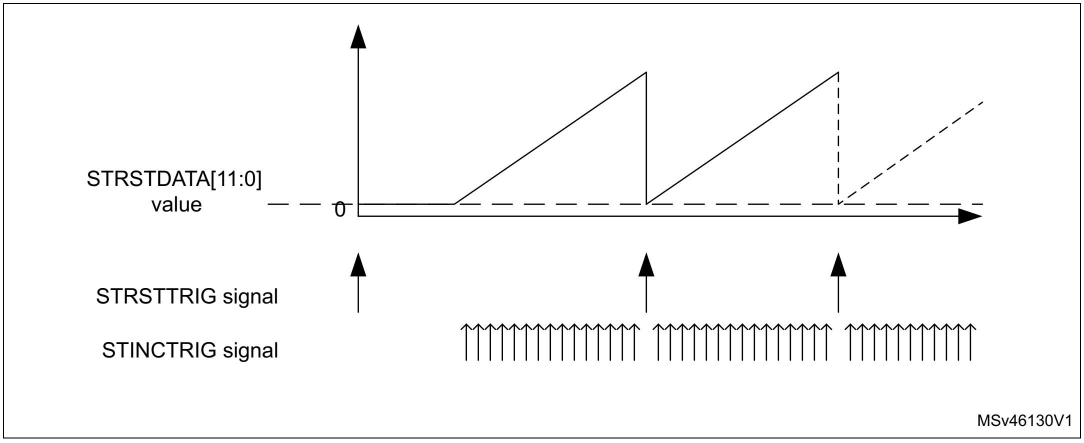
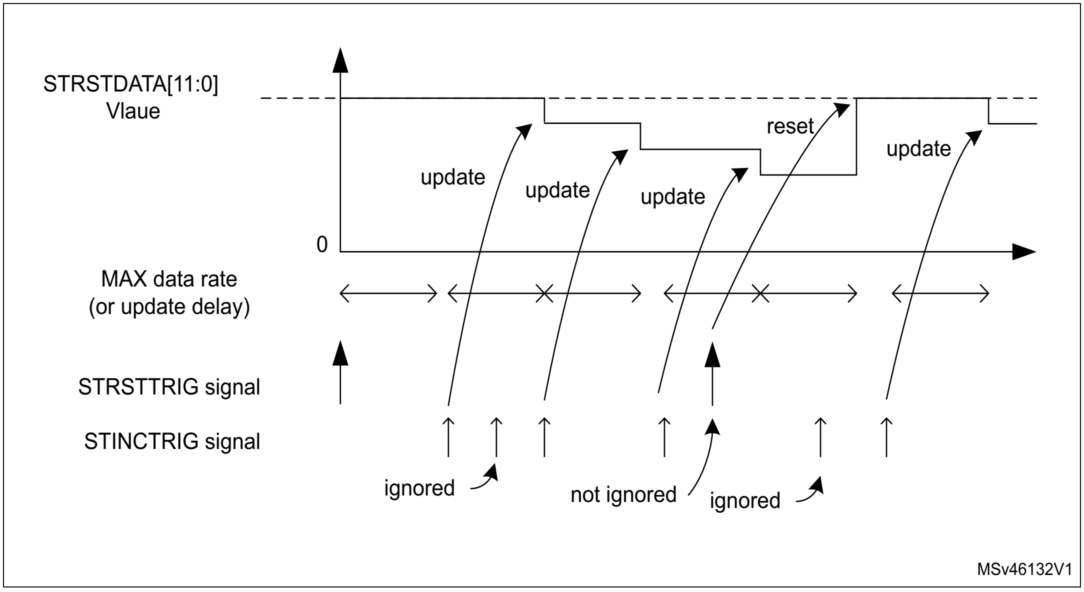


Figure 165. DAC sawtooth wave generation (STDIRx = 1)



The STRSTTRIG signal has higher priority than STINCTRIG. Thetrigger signal cannot be faster than the DAC_DORx update rate defined in the *Table 187: HFSEL description*. If STINCTRIG is asserted faster than the allowed data update rate, the STINCTRIG trigger is ignored. If the STRSTTRIG signal is applied after the STINCTRIG and before DAC_DORx update rate constraints, STRSTTRIG is put on hold. Then, immediately after the data increment, the reset trigger is applied.

Figure 166. DAC sawtooth STINCTRIG and STRSTTRIG priority (STDIR = 0)



22.4.12 DAC channel modes

Each DAC channel can be configured in Normal mode or Sample and hold mode. The output buffer can be enabled to allow 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_OUTy pin interconnections.

To enable the output buffer, the MODEx[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 MODEx[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 or LSE low-speed clock (dac_hold_ck) in addition to the dac_hclk clock, allowing to use the DAC channels in deep low power modes such as Stop mode.

The LSI or LSE 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 TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB 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 THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI/LSE 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/LSE ~32 KHz is selected:

12 cycles are required for sample phase: TSAMPLEx[9:0] = 11,

62 cycles are required for hold phase: THOLDx[9:0] = 62,

and 4 cycles are required for refresh period: TREFRESHx[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 188. 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. Please refer to the product datasheet for product data.

C_{SH} = 100 nF

V_{DDA} = 3.0 V

Sampling phase:

> $t_{SAMP} = 7\ \mu s + (10 * 2000 * 100 * 10^{-9}) = 2.007$ ms
> (where $R_{BON} = 2\ k\Omega$)

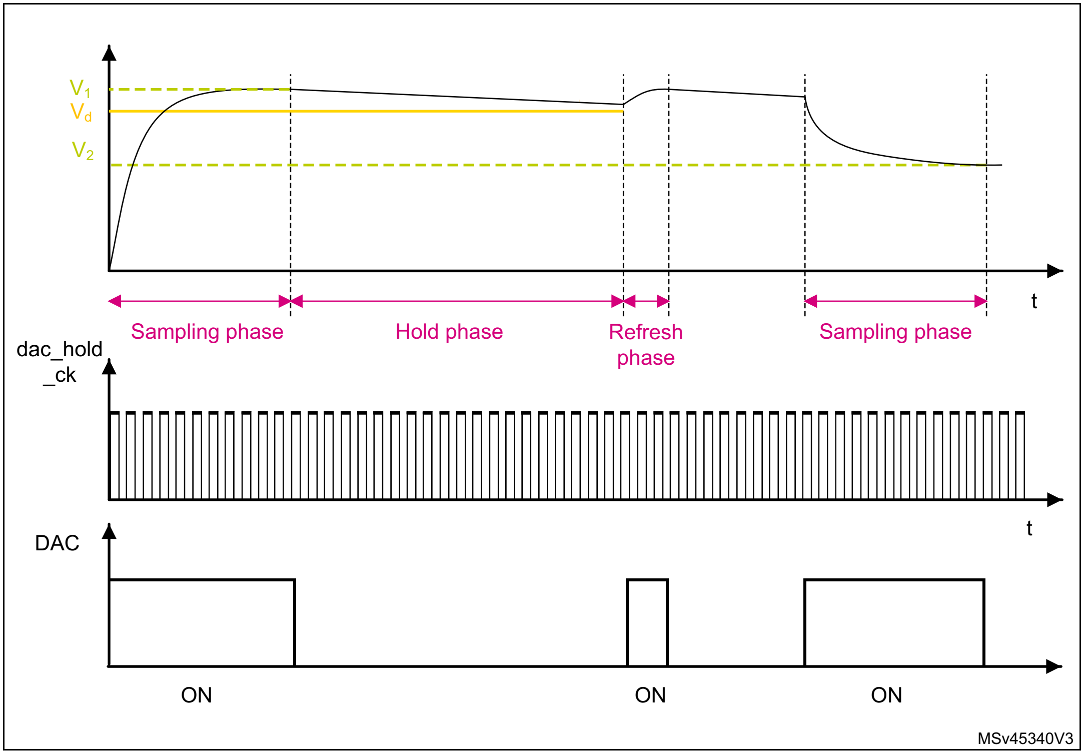
Refresh phase:

> $t_{REFRESH} = 7\ \mu s + (2000 * 100 * 10^{-9}) * \ln(2*10) = 606.1\ \mu s$
> (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 167. DAC Sample and hold mode phase diagram



Like in Normal mode, the Sample and hold mode has different configurations.

To enable the output buffer, MODEx[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, MODEx[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 MODEx[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_DHRx immediately triggers a new sample phase.

Note: The sawtooth wave generation is not supported in Sample and hold mode operation.

Table 189. Channel output modes summary

MODEx[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)
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)

22.4.13 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 channelx is operating with buffer enabled (MODEx[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_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[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 ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 000b or 001b or 100b or 101b.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 00000b.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) If CAL_FLAGx is set to 1, the OTRIMx[4:0] trimming code is found and can be used during *device* operation to compensate the output value, else increment OTRIMx[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 OTRIMx[4:0] bits in a faster way.

The commutation/toggle of CAL_FLAGx bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIMx[4:0] bits in DAC_CCR register.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx 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 OTRIMx[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 CENx bit is set, it is not allowed to set ENx bit.

22.4.14 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

15 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_hclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_hclk 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 two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 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_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 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 two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 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 different trigger sources by setting different values in STRSTTRIGSEL1[3:0], STRSTTRIGSEL2[3:0], STINCTRIGSEL2[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx values for each register.

When a DAC channel1 trigger arrives, the DAC channel1 sawtooth counter updates the DHR1 register and transfers it into DAC_DOR1 (three AHB clock cycles later).

When a DAC channel2 trigger arrives, the DAC channel2 sawtooth counter updates the DHR1 register and transfers it into DAC_DOR1 (three AHB clock cycles later).

Independent trigger with different sawtooth wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure different trigger sources by setting different values in the STRSTTRIGSEL1[23:0], STRSTTRIGSEL2[23:0], STINCTRIGSEL2[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits as 11 and set different STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx value for each register.

When a DAC channel1 trigger arrives, the DAC channel1 sawtooth counter updates the DHR1 register and loads it into DAC_DOR1 (three AHB clock cycles later).

When a DAC channel2 trigger arrives, the DAC channel2 sawtooth counter updates the DHR2 register and loads it into DAC_DOR1 (three AHB clock cycles later).

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_hclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_hclk clock cycles).

Simultaneous trigger with single LFSR generation

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a 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_hclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three AHB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the DAC channel2 triangle counter is updated.

Simultaneous trigger with single sawtooth wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the same trigger source for both DAC channels by setting the same value in STRSTTRIGSEL1[3:0],STRSTTRIGSEL2[3:0], STINCTRIGSEL2[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx value for each register.

When a trigger arrives, the DAC channel1/2 sawtooth counter updates DHR1 and DHR2 registers and loads them into DAC_DOR1/2 (three AHB clock cycles later).

Simultaneous trigger with different sawtooth wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the same trigger source for both DAC channels by setting the same value in STRSTTRIGSEL1[3:0], STRSTTRIGSEL2[3:0], STINCTRIGSEL2[3:0] bits and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits to 11 and set different STRSTDATAx[11:0], STINCDATAx[15:0], STDIRx values for each register.

When a trigger arrives, DAC channel1/2 sawtooth counter updates the DHR1 and DHR2 registers and loads themindependently into DAC_DOR1/2 (three AHB clock cycles later).

22.5 DAC in low-power modes

Table 190. 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.
Stop 0 / Stop 1	the DAC remains active with a static value if the Sample and hold mode is selected using LSI/LSE clock.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

22.6 DAC interrupts

Table 191. 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	DMAUDRx	DMAUDRIEx	Write DMAUDRx = 1	Yes	No	No

22.7 DAC registers

Refer to *Section 1 on page 73* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

22.7.1 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.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3]	TSEL2[2]	TSEL2[1]	TSEL2[0]	TEN2	EN2
	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.	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.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 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 channel2 in Normal operating mode

1: DAC channel2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 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

Note: These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Sawtooth wave generation enabled

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)

These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

0000: SWTRIG2
0001: dac_ch2_trg1
0010: dac_ch2_trg2
...
1111: dac_ch2_trg15

Refer to the trigger selection tables in *Section 22.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).

These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_hclk clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_hclk clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 reg*ister to the DAC_DOR2 register takes only one dac_hclk clock cycle.*

These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

0: DAC channel2 disabled

1: DAC channel2 enabled

Note: These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

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

10: Triangle wave generation enabled

11: Sawtooth 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 22.4.2: 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_hclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_hclk 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_hclk 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

22.7.2 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.	SWTRIG B2	SWTRIG B1
														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.	SWTRIG2	SWTRIG1
														w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **SWTRIGB2**: DAC channel2 software trigger B

This bit is set by software to trigger the DAC in software trigger mode (sawtooth generation) It is cleared by hardware.
0: No trigger
1: Trigger for sawtooth increment

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 16 **SWTRIGB1**: DAC channel1 software trigger B

This bit is set by software to trigger the DAC in software trigger mode (sawtooth generation) It is cleared by hardware.
0: No trigger
1: Trigger for sawtooth increment

Bit 15:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 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_hclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

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_hclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

22.7.3 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.	DACC1DHRB[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.	DACC1DHR[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 **DACC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in Double data mode.

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. They specify 12-bit data for DAC channel1.

22.7.4 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
DACC1DHRB[11:0]												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
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC1DHRB[11:0]**: DAC channel1 12-bit left-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in Double data mode.

Bits 19:16 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.

22.7.5 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
DACC1DHRB[7:0]								DACC1DHR[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 **DACC1DHRB[7:0]**: DAC channel1 8-bit right-aligned data

These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in Double data mode.

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.

22.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

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.	DACC2DHRB[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.	DACC2DHR[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 **DACC2DHRB[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in DMA Double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.

22.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHRB[11:0]												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
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHRB[11:0]**: DAC channel2 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in Double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

22.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Address offset: 0x1C

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
DACC2DHRB[7:0]								DACC2DHR[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 **DACC2DHRB[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software. They specify 8-bit data for DAC channel2 when the DAC operates in Double data mode.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

22.7.9 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.	DACC2DHR[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.	DACC1DHR[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 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

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.

22.7.10 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
DACC2DHR[11:0]												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
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 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.

22.7.11 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
DACC2DHR[7:0]								DACC1DHR[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 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

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.

22.7.12 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.	DACC1DORB[11:0]											
				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.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DORB[11:0]**: DAC channel1 data output

These bits are read-only. They contain data output for DAC channel1 B.

Bits 15:12 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.

22.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DORB[11:0]											
				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.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DORB[11:0]**: DAC channel2 data output

These bits are read-only. They contain data output for DAC channel2 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output

These bits are read-only, they contain data output for DAC channel2.

22.7.14 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
BWST2	CAL_ FLAG2	DMAU DR2	DORST AT2	DAC2R DY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BWST1	CAL_ FLAG1	DMAU DR1	DORST AT1	DAC1R DY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											

Bit 31 **BWST2**: DAC channel2 busy writing sample time flag

This bit is systematically set just after Sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 30 **CAL_FLAG2**: DAC channel2 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

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 29 **DMAUDR2**: DAC channel2 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 channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 28 **DORSTAT2**: DAC channel2 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in Double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 27 **DAC2RDY**: DAC channel2 ready status bit

This bit is set and cleared by hardware.

0: DAC channel2 is not yet ready to accept the trigger nor output data

1: DAC channel2 is ready to accept the trigger or output data

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

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/LSE 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 **DORSTAT1**: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in Double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Bit 11 **DAC1RDY**: DAC channel1 ready status bit

This bit is set and cleared by hardware.

0: DAC channel1 is not yet ready to accept the trigger nor output data

1: DAC channel1 is ready to accept the trigger or output data

Bits 10:0 Reserved, must be kept at reset value.

22.7.15 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.	OTRIM2[4:0]				
											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.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

22.7.16 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.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
						rw	rw						rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HFSEL1	HFSEL0	Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
rw	rw					rw	rw						rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SINFORMAT2**: Enable signed format for DAC channel2

This bit is set and cleared by software.

0: Input data is in unsigned format

1: Input data is in signed format (2’s complement). The MSB bit represents the sign.

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 24 **DMADOUBLE2**: DAC channel2 DMA double data mode

This bit is set and cleared by software.

0: DMA Normal mode selected

1: DMA Double data mode selected

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

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

Bits 18:16 **MODE2[2:0]:** DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

– DAC channel2 in Normal mode

000: DAC channel2 is connected to external pin with Buffer enabled

001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled

010: DAC channel2 is connected to external pin with buffer disabled

011: DAC channel2 is connected to on chip peripherals with Buffer disabled

– DAC channel2 in Sample and hold mode

100: DAC channel2 is connected to external pin with Buffer enabled

101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel2 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN2 = 0.

Refer to Section 22.3: DAC implementation for the availability of DAC channel2.

Bits 15:14 **HFSEL[1:0]**: High frequency interface mode selection

00: High frequency interface mode disabled

01: High frequency interface mode compatible to AHB>80 MHz enabled

10: High frequency interface mode compatible to AHB>160 MHz enabled

11: Reserved

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1

This bit is set and cleared by software.

0: Input data is in unsigned format

1: Input data is in signed format (2’s complement). The MSB bit represents the sign.

Bit 8 **DMADOUBLE1**: DAC channel1 DMA double data mode

This bit is set and cleared by software.

0: DMA Normal mode selected

1: DMA Double data mode selected

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.

22.7.17 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/LSE clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

22.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

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.	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.	TSAMPLE2[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 **TSAMPLE2[9:0]:** DAC channel2 sample time (only valid in Sample and hold mode)

These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2 = 1, the write operation is ignored.

Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

22.7.19 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.	THOLD2[9:0]									
						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.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **THOLD2[9:0]:** DAC channel2 hold time (only valid in Sample and hold mode).

Hold time = (THOLD[9:0]) x LSI/LSE clock period

Note: This register can be modified only when EN2 = 0.

These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

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/LSE 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 ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

22.7.20 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.	TREFRESH2[7:0]							
								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.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TREFRESH2[7:0]:** DAC channel2 refresh time (only valid in Sample and hold mode)

Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period

Note: This register can be modified only when EN2 = 0.

These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

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/LSE 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 ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

22.7.21 DAC channel1 sawtooth register (DAC_STR1)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STINCDATA1[15:0]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	STDIR 1	STRSTDATA1[11:0]											
			rw	rw											

Bits 31:16 **STINCDATA1[15:0]:** DAC channel1 sawtooth increment value (12.4 bit format)

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **STDIR1:** DAC channel1 sawtooth direction setting
This bit is written by software to select the direction of Sawtooth step direction
0: Decrement
1: Increment

Bits 11:0 **STRSTDATA1[11:0]:** DAC channel1 sawtooth reset value

22.7.22 DAC channel2 sawtooth register (DAC_STR2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STINCDATA2[15:0]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	STDIR 2	STRSTDATA2[11:0]											
			rw	rw											

Bits 31:16 **STINCDATA2[15:0]:** DAC channel2 Sawtooth increment value (12.4 bit format)

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **STDIR2:** DAC channel2 sawtooth direction setting
This bit is written by software to select the direction of sawtooth step direction
0: Decrement
1: Increment

Bits 11:0 **STRSTDATA2[11:0]:** DAC channel2 sawtooth reset value

22.7.23 DAC sawtooth mode register (DAC_STMODR)

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.	STINCTRIGSEL2[3:0]				Res.	Res.	Res.	Res.	STRSTTRIGSEL2[3:0]			
				rw								rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	STINCTRIGSEL1[3:0]				Res.	Res.	Res.	Res.	STRSTTRIGSEL1[3:0]			
				rw								rw			

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **STINCTRIGSEL2[3:0]:** DAC channel2 sawtooth increment trigger selection

Refer to the trigger selection tables in *Section 22.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

0000: SWTRIGB2

0001: dac_inc_ch2_trg1

....

1111: dac_inc_ch2_trg15

Note: These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 19:16 **STRSTTRIGSEL2[3:0]:** DAC channel2 sawtooth reset trigger selection

Refer to the trigger selection tables in *Section 22.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

0000: SWTRIGB2

0001: dac_ch2_trg1

....

1111: dac_ch2_trg15

The mapping is the same as for TSEL2[3:0].

Note: These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **STINCTRIGSEL1[3:0]:** DAC channel1 sawtooth increment trigger selection

Refer to the trigger selection tables in *Section 22.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

0000: SWTRIGB1

0001: dac_inc_ch1_trg1

....

1111: dac_inc_ch1_trg15

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **STRSTTRIGSEL1[3:0]:** DAC channel1 sawtooth reset trigger selection

Refer to the trigger selection tables in *Section 22.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

0000: SWTRIGB1

0001: dac_ch1_trg1

....

1111: dac_ch1_trg15

The mapping is the same as for TSEL1[3:0].

22.7.24 DAC register map

Table 192 summarizes the DAC registers.

Table 192. 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.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL23	TSEL22	TSEL21	TSEL20	TEN2	EN2	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.	SWTRIGB2	SWTRIGB1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value															0	0															0	0
0x08	DAC_DHR12R1	Res.	Res.	Res.	Res.	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[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
0x0C	DAC_DHR12L1	DACC1DHRB[11:0]												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					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.	DACC1DHRB[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	DAC_DHR12R2	Res.	Res.	Res.	Res.	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[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
0x18	DAC_DHR12L2	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]												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	0	0	0	0				
0x1C	DAC_DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHRB[7:0]								DACC2DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DAC_DHR12RD	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[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
0x24	DAC_DHR12LD	DACC2DHR[11:0]												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					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.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	DACC1DORB[11:0]												Res.	Res.	Res.	Res.	DACC1DOR[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
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	DACC2DORB[11:0]												Res.	Res.	Res.	Res.	DACC2DOR[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

Table 192. 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
0x34	**DAC_SR**	BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0												0	0	0	0	0											
0x38	**DAC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X
0x3C	**DAC_MCR**	Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			HFSEL1	HFSEL0	Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value							0	0						0	0	0	0	0					0	0						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
0x44	**DAC_SHSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	**DAC_SHHR**	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	1							0	0	0	0	0	0	0	0	0	1
0x4C	**DAC_SHRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1
0x58	DAC_STR1	STINCDATA1[15:0]																Res.	Res.	Res.	Res.	STRSTDATA1[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	0
0x5C	DAC_STR2	STINCDATA2[15:0]																Res.	Res.	Res.	Res.	STRSTDATA2[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	0
0x60	DAC_STMODR	Res.	Res.	Res.	Res.	STINCTRIG SEL2[3:0]				Res.	Res.	Res.	Res.	STRSTTRIG SEL2[3:0]				Res.	Res.	Res.	Res.	STINCTRIG SEL1[3:0]				Res.	Res.	Res.	Res.	STRSTTRIG SEL1[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0

Refer to *Section 2.2 on page 81* for the register boundary addresses.

23 Voltage reference buffer (VREFBUF)

23.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for ADCs, DACs 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).

23.2 VREFBUF functional description

The internal voltage reference buffer supports three voltages[a], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 00: around 2.048 V.
- VRS = 01: around 2.5 V.
- VRS = 10: around 2.90 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 193. 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: – VREFBUF buffer off – V_{REF+} pin floating. The voltage is held 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.

23.3 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

23.4 VREFBUF registers

23.4.1 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.	VRS[1:0]		VRR	Res.	HIZ	ENVR
										rw	rw	r		rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **VRS[1:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

00: Voltage reference set to 2.048 V
01: Voltage reference set to 2.5 V
10: Voltage reference set to 2.90 V
11: Reserved

Note: The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

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 Reserved, must be kept at reset value.

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 193: 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.

23.4.2 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

The TRIM code is a 6-bit unsigned data (minimum 000000, maximum 111111) that is set and updated according the mechanism described below.

Reset:

TRIM[5:0] is automatically initialized with the VRS = 0 trimming value stored in the Flash memory during the production test.

VRS change:

TRIM[5:0] is automatically initialized with the trimming value (corresponding to VRS setting) stored in the Flash memory during the production test.

Write in TRIM[5:0]:

User can modify the TRIM[5:0] with an arbitrary value. This is permanently disabling the control of the trimming value with VRS (until the device is reset).

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

23.4.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 194. 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.	VRR	VRS[1:0]		HIZ	ENVR
	Reset value																												0	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 2.2 on page 81* for the register boundary addresses.

24 Comparator (COMP)

24.1 COMP introduction

The device embeds up to seven ultra-fast analog comparators.

The comparators can be used for a variety of functions including:

- 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.

24.2 COMP main features

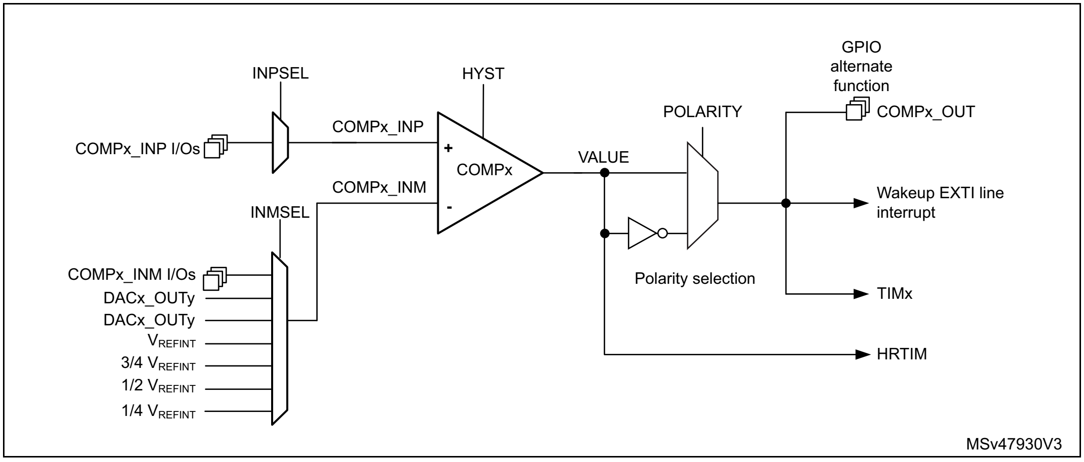
- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - DAC channels
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by a scaler (buffered voltage divider)
- Programmable hysteresis
- Output redirection to I/Os or to timer inputs for triggering break events for fast PWM shutdowns
- Output blanking for immunity to switching noise
- Per-channel interrupt generation with wake-up from Sleep and Stop modes

24.3 COMP functional description

24.3.1 COMP block diagram

The block diagram of one comparator channel front-end is shown in *Figure 168: Comparator block diagram*.

Figure 168. Comparator block diagram



24.3.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The comparator output can be connected to the I/Os using the alternate function channel given in "Alternate function mapping" table in the datasheet.

The output can also be internally redirected to a variety of timer input 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 measures

It is possible to have the comparator output simultaneously redirected internally and externally.

Table 195. COMPx non-inverting input assignment

INPSEL	COMP1_ INP	COMP2_ INP	COMP3_ INP	COMP4_ INP	COMP5_ INP	COMP6_ INP	COMP7_ INP
0	PA1	PA7	PA0	PB0	PB13	PB11	PB14
1	PB1	PA3	PC1	PE7	PD12	PD11	PD14

Table 196. COMPx inverting input assignment

INMSEL [2:0]	COMP1_ INM	COMP2_ INM	COMP3_ INM	COMP4_ INM	COMP5_ INM	COMP6_ INM	COMP7_ INM
000	1/4 V_{REFINT}						
001	1/2 V_{REFINT}						
010	3/4 V_{REFINT}						
011	V_{REFINT}						
100	DAC3_CH1	DAC3_CH2	DAC3_CH1	DAC3_CH2	DAC4_CH1	DAC4_CH2	DAC4_CH1
101	DAC1_CH1	DAC1_CH2	DAC1_CH1	DAC1_CH1	DAC1_CH2	DAC2_CH1	DAC2_CH1
110	PA4	PA5	PF1	PE8	PB10	PD10	PD15
111	PA0	PA2	PC0	PB2	PD13	PB15	PB12

24.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the APB2 clock.

There is no COMP-dedicated clock enable control bit 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 of APB clock. This allows the comparator to work even in Stop mode.*

24.3.4 COMP LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

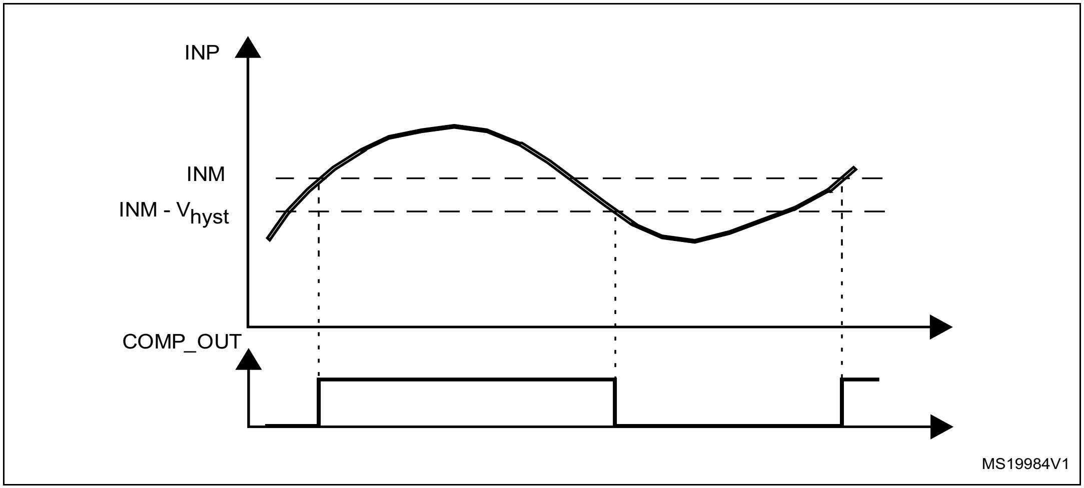
Once the programming is completed, the COMPx LOCK bit can be set. This causes the whole register to become read-only, including the COMPx LOCK bit.

The write protection can only be removed by an MCU reset.

24.3.5 COMP hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions with noisy input signals. It is non-symmetrical and only acting to falling edge of the comparator output. The internal hysteresis function can be disabled so as to set the amount of hysteresis with external components, which can be useful for example when exiting a low-power mode.

Figure 169. Comparator hysteresis



24.3.6 COMP output blanking

The purpose of the blanking function is to prevent the current regulation from tripping upon short current spikes at the beginning of PWM period (typically the recovery current in power switch anti-parallel diodes). This goes through setting a dead window defined with a timer output compare signal. The blanking source is selected individually per comparator channel by software through BLANKSEL[2:0] bitfield of corresponding COMP_CxCSR register, as shown in *Table 197: Blanking sources*. The inverted blanking signal is logical AND-ed with the comparator stage output to produce the comparator channel x output. See the example provided in the following figure.

Figure 170. Comparator output blanking

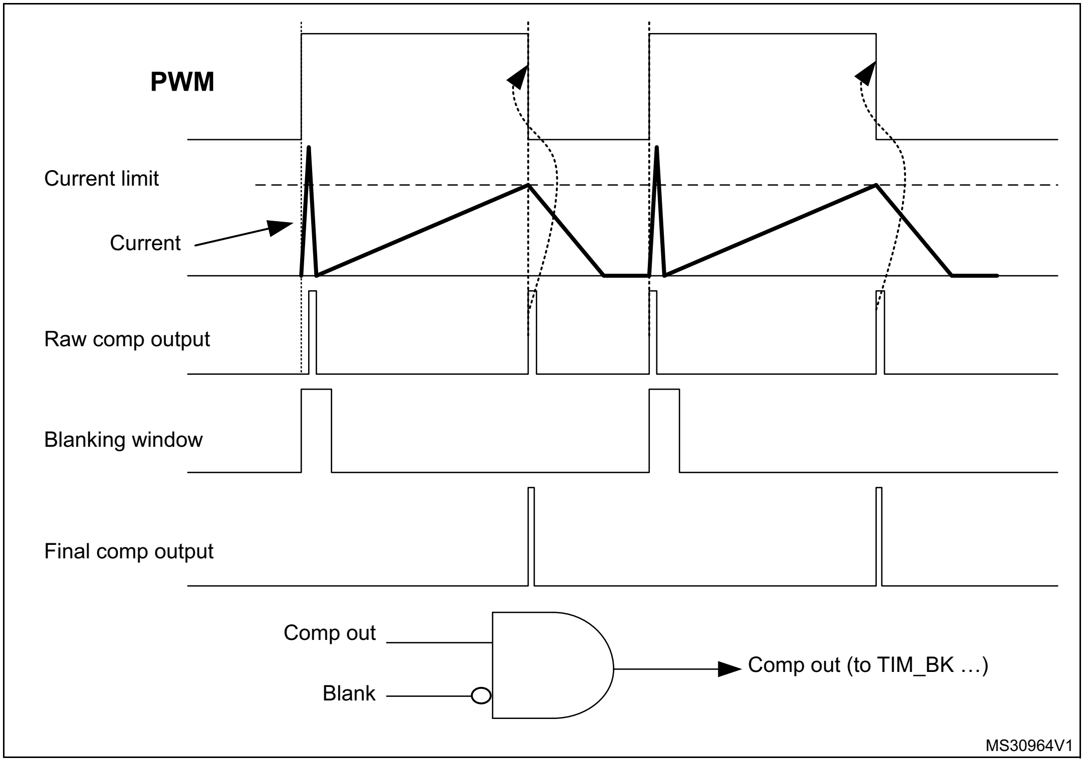


Table 197. Blanking sources

BLANKSEL [2:0]	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7
001	TIM1_OC5	TIM1_OC5	TIM1_OC5	TIM3_OC4	TIM2_OC3	TIM8_OC5	TIM1_OC5
010	TIM2_OC3	TIM2_OC3	TIM3_OC3	TIM8_OC5	TIM8_OC5	TIM2_OC4	TIM8_OC5
011	TIM3_OC3	TIM3_OC3	TIM2_OC4	TIM15_OC1	TIM3_OC3	TIM15_OC2	TIM3_OC3
100	TIM8_OC5	TIM8_OC5	TIM8_OC5	TIM1_OC5	TIM1_OC5	TIM1_OC5	TIM15_OC2
101	TIM20_OC5	TIM20_OC5	TIM20_OC5	TIM20_OC5	TIM20_OC5	TIM20_OC5	TIM20_OC5
110	TIM15_OC1	TIM15_OC1	TIM15_OC1	TIM15_OC1	TIM15_OC1	TIM15_OC1	TIM15_OC1
111	TIM4_OC3	TIM4_OC3	TIM4_OC3	TIM4_OC3	TIM4_OC3	TIM4_OC3	TIM4_OC3

24.4 COMP low-power modes

Table 198. Comparator behavior in low-power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Low-power run	No effect.

Table 198. Comparator behavior in low-power modes (continued)

Mode	Description
Low-power sleep	No effect. COMP interrupts cause the device to exit the Low-power sleep mode.
Stop	No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode.
Standby, Shutdown	The COMP registers are powered down and must be reinitialized after exiting Standby or Shutdown mode.

24.5 COMP interrupts

The comparator outputs are internally connected to the Extended interrupts and events controller. Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit Sleep and Stop low-power modes.

Refer to Interrupt and events section for more details.

To enable the COMPx interrupt, it is required to follow this sequence:

1. Configure and enable the EXTI line corresponding to the COMPx output event in interrupt mode and select sensitivity to rising edge, falling edge or to both edges
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines
3. Enable COMPx

Interrupt events are flagged through COMP_CxCSR flags.

24.6 COMP registers

24.6.1 Comparator x control and status register (COMP_CxCSR)

For x = 1 through 7, the COMP_CxCSR register contains all bits and flags related to the comparator x.

Address offset: 4(x-1), where x = 1 to 7

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.						SCALEN	BRGEN	BLANKSEL[2:0]			HYST		
rw	r							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	Res.						INPSEL	INMSEL[2:0]				Res.			EN
rw							rw	rw	rw	rw	rw				rw

Bit 31 **LOCK:** COMP_CxCSR register lock

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator x control register COMP_CxCSR[31:0]. When locked, all control bits and flags can be read only but not written. When unlocked, the control bits can also be written by software.

0: Unlock

1: Lock

Bit 30 **VALUE:** Comparator x output status

This read-only flag reflects the level of the comparator x output before the polarity selector and blanking, as indicated in *Figure 168*.

Bits 29:24 Reserved, must be kept at reset value

Bit 23 **SCALEN:** V_{REFINT} scaler enable

This bit controlled by software enables the operation of V_{REFINT} scaler at the inverting input of all comparator. To disable the V_{REFINT} scaler, SCALEN bits of all COMP_CxCSR registers must be set to Disable state. When the V_{REFINT} scaler is disabled, the 1/4 V_{REFINT}, 1/2 V_{REFINT}, 3/4 V_{REFINT} and V_{REFINT} inputs of the multiplexer should not be selected.

0: Disable

1: Enable

Bit 22 **BRGEN:** V_{REFINT} scaler resistor bridge enable

This bit controlled by software enables the operation of resistor bridge in the V_{REFINT} scaler. To disable the resistor bridge, BRGEN bits of all COMP_CxCSR registers must be set to Disable state. When the resistor bridge is disabled, the 1/4 V_{REFINT}, 1/2 V_{REFINT}, and 3/4 V_{REFINT} inputs of the input selector receive V_{REFINT} voltage.

0: Disable

1: Enable

Bits 21:19 **BLANKSEL[2:0]:** Comparator x blanking signal select

This bitfield controlled by software selects the blanking signal for comparator channel x, as shown in *Table 197: Blanking sources*.

Bits 18:16 **HYST[2:0]:** Comparator x hysteresis

This bitfield controlled by software selects the hysteresis of the comparator x:

000: No hysteresis

001: 10mV hysteresis

010: 20mv hysteresis

011: 30mV hysteresis

100: 40mV hysteresis

101: 50mV hysteresis

110: 60mV hysteresis

111: 70mV hysteresis

Bit 15 **POL:** Comparator x polarity

This bit controlled by software selects the comparator x output polarity:

0: Non-inverted

1: Inverted

Bits 14:9 Reserved, must be kept at reset value

Bit 8 **INPSEL**: Comparator x signal select for non-inverting input

This bitfield controlled by software selects the signal for the non-inverting input COMPx_INP of the comparator x, as shown in *Table 195: COMPx non-inverting input assignment*.

Bits 7:4 **INMSEL[3:0]**: Comparator x signal select for inverting input

This bitfield controlled by software selects the signal for the inverting input COMPx_INM of the comparator x, as shown in *Table 196: COMPx inverting input assignment*.

Bits 3:1 Reserved, must be kept at reset value

Bit 0 **EN:** Comparator x enable

This bit controlled by software enables the operation of comparator x:

0: Disable

1: Enable

24.6.2 COMP register map

The following table summarizes the comparator registers.

The comparator registers share SYSCFG peripheral register base addresses.

Table 199. COMP 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	**COMP_C1CSR**	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	BLANKSEL[2:0]			HYST[3:0]			POL	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL.	Res.	INMSEL[3:0]			Res.	Res.	Res.	EN
	Reset value	0	0							0	0	0	0	0		0	0	0							0		0	0	0				0
0x04	**COMP_C2CSR**	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	BLANKSEL[2:0]			Res.	HYST		POL	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL.	Res.	INMSEL[2:0]			Res.	Res.	Res.	EN
	Reset value	0	0							0	0	0	0	0		0	0	0							0		0	0	0				0
0x08	**COMP_C3CSR**	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	BLANKSEL[2:0]			Res.	HYST		POL	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL.	Res.	INMSEL[2:0]			Res.	Res.	Res.	EN
	Reset value	0	0							0	0	0	0	0		0	0	0							0		0	0	0				0
0x0C	**COMP_C4CSR**	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	BLANKSEL[2:0]			Res.	HYST		POL	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL.	Res.	INMSEL[2:0]			Res.	Res.	Res.	EN
	Reset value	0	0							0	0	0	0	0		0	0	0							0		0	0	0				0
0x10	**COMP_C5CSR**	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	BLANKSEL[2:0]			Res.	HYST		POL	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL.	Res.	INMSEL[2:0]			Res.	Res.	Res.	EN
	Reset value	0	0							0	0	0	0	0		0	0	0							0		0	0	0				0

Table 199. COMP 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
0x14	COMP_C6CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	BLANKSEL[2:0]			Res.	HYST		POL	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL.	Res.	INMSEL[2:0]			Res.	Res.	Res.	EN
	Reset value	0	0							0	0	0	0	0		0	0	0							0		0	0	0				0
0x18	COMP_C7CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	BLANKSEL[2:0]			Res.	HYST		POL	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL.	Res.	INMSEL[2:0]			Res.	Res.	Res.	EN
	Reset value	0	0							0	0	0	0	0		0	0	0							0		0	0	0				0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

25 Operational amplifiers (OPAMP)

25.1 Introduction

The devices embed six operational amplifiers with two inputs and one output each. The three I/Os can be connected to the external pins, thus enabling any type of external interconnections. The operational amplifiers can be configured internally as a follower, as an amplifier with a non-inverting gain ranging from 2 to 64 or with inverting gain ranging from -1 to -63.

The positive input can be connected to the internal DAC.

The output can be connected to the internal ADC.

25.2 OPAMP main features

- Rail-to-rail input and output voltage range
- Low input bias current
- Low input offset voltage
- high frequency gain bandwidth
- High-speed mode to achieve a better slew rate

Note: Refer to the product datasheet for detailed OPAMP characteristics.

25.3 OPAMP functional description

The OPAMP has several modes.

Each OPAMP can be individually enabled, when disabled the output is high-impedance.

When enabled, it can be in calibration mode, all input and output of the OPAMP are then disconnected, or in functional mode.

In functional mode the inputs and output of the OPAMP are connected as described in *Section 25.3.4: Signal routing*.

25.3.1 OPAMP output redirection to internal ADC channels

Operational amplifier output can be redirected internally to an ADC channel by setting OPAINTOEN bit in OPAMPx_CSR register. In this case, the GPIO on which is mapped given OPAMPx_VOUT output is free and can be used for another purpose. ADC can measure OPAMPx_VOUT voltage internally. See *Figure 200: Operational amplifier possible connection* for assignment between OPAMPx internal outputs and ADC channels.

25.3.2 OPAMP reset and clocks

The OPAMP clock provided by the clock controller is synchronized with the PCLK2 (APB2 clock). There is no clock enable control bit provided in the RCC controller. To use a clock source for the OPAMP, the SYSCFG clock enable control bit must be set in the RCC controller.

The bit OPAEN enables and disables the OPAMP operation. The OPAMP registers configurations must be changed before enabling the OPAEN bit in order to avoid spurious effects on the output.

When the output of the operational amplifier is no more needed the operational amplifier can be disabled to save power. All the configurations previously set (including the calibration) are maintained while OPAMP is disabled.

25.3.3 Initial configuration

The default configuration of the operational amplifier is a functional mode where the three input/outputs are connected to external pins. In the default mode the operational amplifier uses the factory trimming values for its offset calibration. The trimming values can be adjusted, see *Section 25.3.7: Calibration* for changing the trimming values. The default configuration uses the normal mode, which provides the standard performance. The bit OPAHSM can be set in order to switch the operational amplifier to high-speed mode for a better slew rate. Both normal and high-speed mode characteristics are defined in the datasheet.

As soon as the OPAEN bit in OPAMPx_CSR register is set, the operational amplifier is functional. The two input pins and the output pin are connected as defined in *Section 25.3.4: Signal routing* and the default connection settings can be changed.

Note: The inputs and output pins must be configured in analog mode (default state) in the corresponding GPIOx_MODER register.

25.3.4 Signal routing

The routing for the operational amplifier pins is determined by OPAMPx_CSR and OPAMPx_TCMR registers.

The connections of the six operational amplifiers (OPAMPx, x = 1...6) are described in the table below.

Table 200. Operational amplifier possible connection

Signal	Pin	Internal	Comment
OPAMP1_VINM	PA3 (VINM0) PC5 (VINM1)	OPAMP1_VOUT or PGA	Controlled by bits PGA_GAIN and VM_SEL
OPAMP1_VINP	PA1 (VINP0) PA3 (VINP1) PA7 (VINP2)	DAC3_CH1	controlled by bit VP_SEL.
OPAMP1_VOUT	PA2	ADC1_IN3 ADC1_IN13[1]	The pin is connected when the OPAMP is enabled and OPAMP internal output is disabled. The ADC input is controlled by ADC
OPAMP2_VINM	PA5 (VINM0) PC5 (VINM1)	OPAMP2_VOUT or PGA	controlled by bits PGA_GAIN and VM_SEL
OPAMP2_VINP	PA7 (VINP0) PB14 (VINP1) PB0 (VINP2) PD14 (VINP3)	-	controlled by bit VP_SEL

Table 200. Operational amplifier possible connection (continued)

Signal	Pin	Internal	Comment
OPAMP2_VOUT	PA6	ADC2_IN3 ADC2_IN16[1]	The pin is connected when the OPAMP is enabled and OPAMP internal output is disabled. The ADC input is controlled by ADC
OPAMP3_VINM	PB2 (VINM0) PB10 (VINM1)	OPAMP3_VOUT or PGA	Controlled by bits PGA_GAIN and VM_SEL
OPAMP3_VINP	PB0 (VINP0) PB13 (VINP1) PA1 (VINP2)	DAC3_CH2[2][3]	Controlled by bit VP_SEL
OPAMP3_VOUT	PB1	ADC3_IN1[2][3]/ADC1_IN12 ADC2_IN18[1]/ADC3_IN13[1][2][3]	The pin is connected when the OPAMP is enabled and OPAMP internal output is disabled. The ADC input is controlled by ADC
OPAMP4_VINM	PB10 (VINM0) PD8 (VINM1)	OPAMP4_VOUT or PGA	Controlled by bits PGA_GAIN and VM_SEL
OPAMP4_VINP	PB13 (VINP0) PD11 (VINP1) PB11 (VINP2)	DAC4_CH1	Controlled by bit VP_SEL
OPAMP4_VOUT	PB12	ADC4_IN3/ADC1_IN11 ADC5_IN5[1]	The pin is connected when the OPAMP is enabled and OPAMP internal output is disabled. The ADC input is controlled by ADC.
OPAMP5_VINM	PB15 (VINM0) PA3 (VINM1)	OPAMP5_VOUT or PGA	Controlled by bits PGA_GAIN and VM_SEL.
OPAMP5_VINP	PB14 (VINP0) PD12 (VINP1) PC3 (VINP2)	DAC4_CH2	Controlled by bit VP_SEL.
OPAMP5_VOUT	PA8	ADC5_IN1 ADC5_IN3[1]	The pin is connected when the OPAMP is enabled and OPAMP internal output is disabled. The ADC input is controlled by ADC.
OPAMP6_VINM	PA1 (VINM0) PB1 (VINM1)	OPAMP6_VOUT or PGA	Controlled by bits PGA_GAIN and VM_SEL.
OPAMP6_VINP	PB12 (VINP0) PD9 (VINP1) PB13 (VINP2)	DAC3_CH1	Controlled by bit VP_SEL.
OPAMP6_VOUT	PB11	ADC12_IN14 ADC4_IN17[1][2] ADC3_IN17[1][3]	The pin is connected when the OPAMP is enabled and OPAMP internal output is disabled. The ADC input is controlled by ADC.

1. This ADC channel is connected internally to the OPAMPx_VOUT when OPAINTOEN bit is set. In this case, the I/O on which the OPAMPx_VOUT is available, can be used for another purpose.
2. For category 3 devices only.
3. For category 4 devices only.
4. OPAMP4/5 are supported in category 3 devices only.
5. OPAMP6 is supported only in category 3 and category 4 devices.

25.3.5 OPAMP modes

The operational amplifier inputs and outputs are all accessible on terminals. The amplifiers can be used in multiple configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

Note: The impedance of the signal must be maintained below a level which avoids the input leakage to create significant artifacts (due to a resistive drop in the source). Please refer to the electrical characteristics section in the datasheet for further details.

Standalone mode (external gain setting mode)

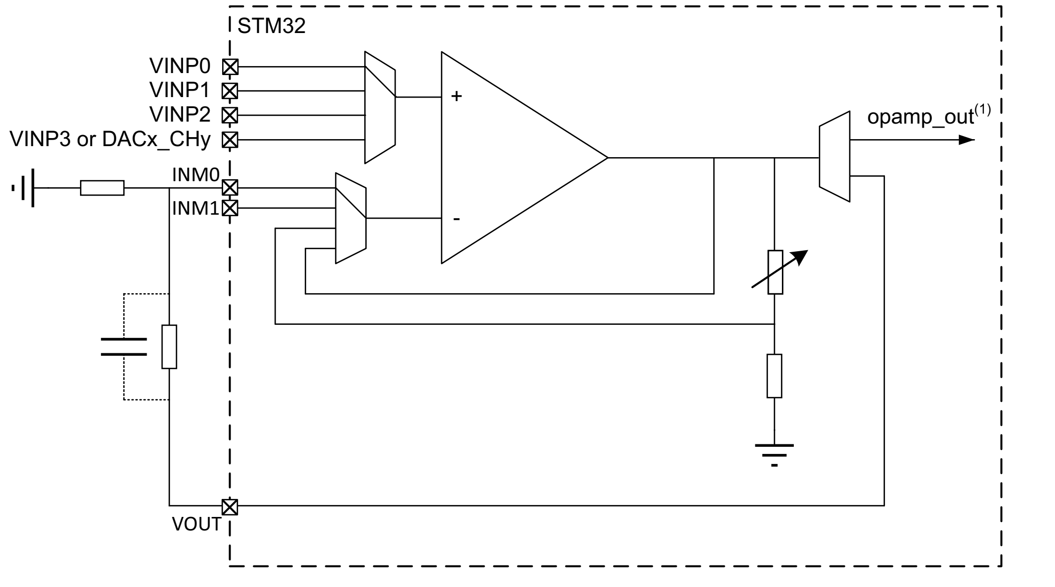
The procedure to use the OPAMP in standalone mode is presented hereafter.

Starting from the default value of OPAMPx_CSR, and the default state of GPIOx_MODER, as soon as the OPAEN bit is set, the two input pins and the output pin are connected to the operational amplifier.

This default configuration uses the factory trimming values and operates in normal mode (highest performance). The behavior of the OPAMP can be changed as follows:

- OPAHSM can be set to “operational amplifier high-speed” mode in order to have high slew rate.
- USERTRIM can be set to modify the trimming values for input offsets.

Figure 171. Standalone mode: external gain setting mode



1. *opamp_out can be redirected internally to an ADC channel by setting OPAINTOEN bit. In this case, the I/O on which is mapped the OPAMPx_VOUT is free and can be used for another purpose.*

MSv48039V1

Follower configuration mode

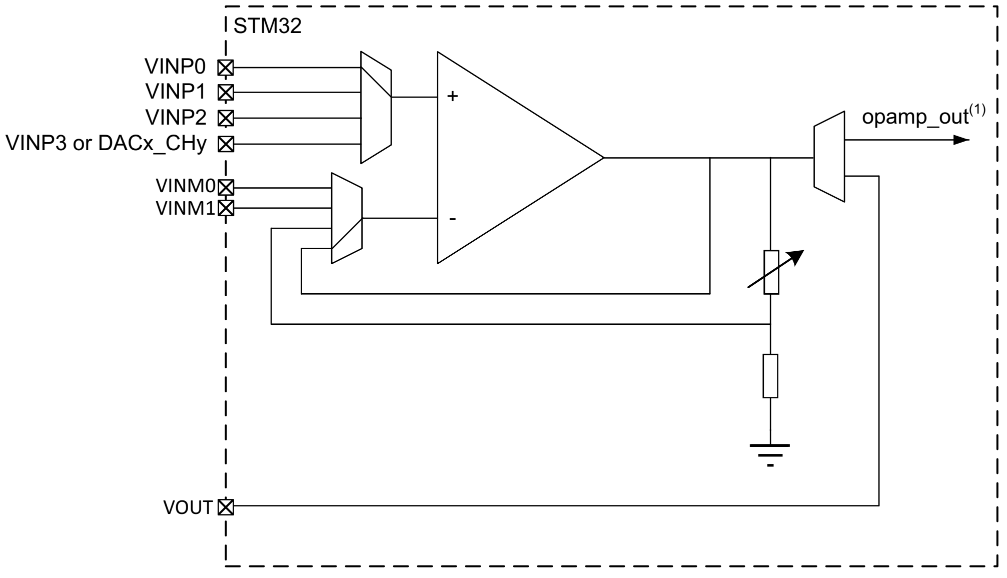
The procedure to use the OPAMP in follower mode is presented hereafter.

- configure VM_SEL bits as “opamp_out connected to OPAMPx_VINM input”, 11
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”, 00
- As soon as the OPAEN bit is set, the voltage on pin OPAMPx_VINP is buffered to pin OPAMPx_VOUT.

Note: The pin corresponding to OPAMPx_VINM is free for another usage.

The signal on the OPAMP output is also seen as an ADC input. As a consequence, the OPAMP configured in follower mode can be used to perform impedance adaptation on input signals before feeding them to the ADC input, assuming the input signal frequency is compatible with the operational amplifier gain bandwidth specification.

Figure 172. Follower configuration



1. *opamp_out can be redirected internally to an ADC channel by setting OPAINTOEN bit. In this case, the I/O on which is mapped the OPAMPx_VOUT is free and can be used for another purpose.*

MSv48040V1

Programmable gain amplifier mode

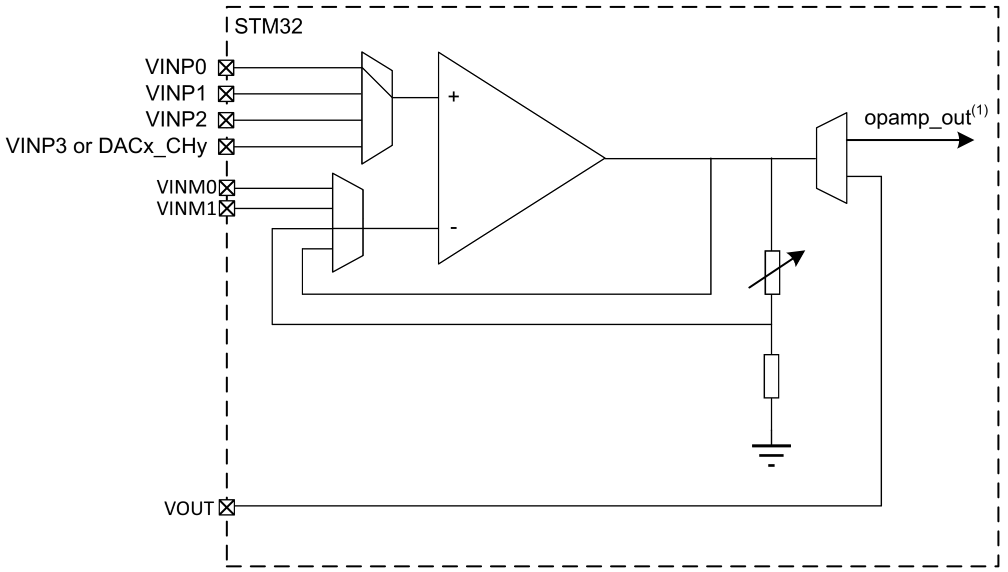
The procedure to use the OPAMP as programmable gain amplifier is presented hereafter.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “internal Gain 2, 4, 8, 16,32, or 64”, 00000 to 00101
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”, 00

  As soon as the OPAEN bit is set, the voltage on pin OPAMPx_VINP is amplified by the selected gain and visible on pin OPAMPx_VOUT.

Note: To avoid saturation, the input voltage must stay below V_{DDA} divided by the selected gain.

Figure 173. PGA mode, internal gain setting (x2/x4/x8/x16/x32/x64), inverting input not used



1. *opamp_out can be redirected internally to an ADC channel by setting OPAINTOEN bit. In this case, the I/O on which is mapped the OPAMPx_VOUT is free and can be used for another purpose.*

MSv48041V1

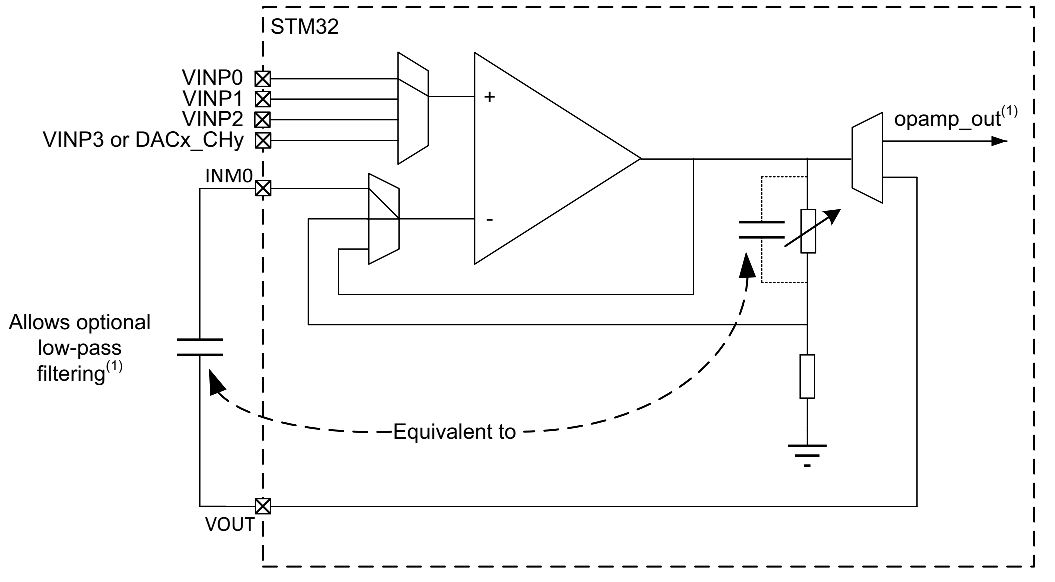
Programmable gain amplifier mode with external filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal, with an external filtering, is presented hereafter.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “internal Gain 2, 4, 8, 16, 32 or 64 with filtering on VINM0”, 10000 to 10101
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

  Any external connection on VINM can be used in parallel with the internal PGA, for example a capacitor can be connected between opamp_out and VINM for filtering purpose (see datasheet for the value of resistors used in the PGA resistor network).

Figure 174. PGA mode, internal gain setting (x2/x4/x8/x16/x32/x64), inverting input used for filtering



1. opamp_out can be redirected internally to an ADC channel by setting OPAINTOEN bit. In this case, the I/O on which is mapped the OPAMPx_VOUT is free and can be used for another purpose.

MSv48042V1

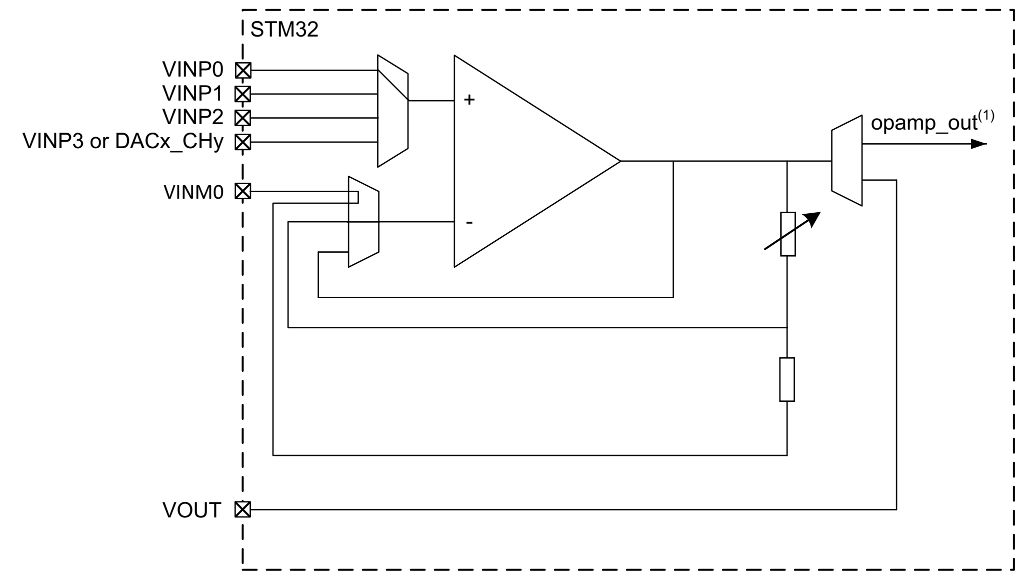
1. The gain depends on the cut-off frequency.

Programmable gain amplifier, non-inverting with external bias or inverting mode

The procedure to use the OPAMP to amplify the amplitude of an input signal with bias voltage for non-inverting mode or inverting mode.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “Inverting gain = -1, -3, -7, -15, -31, -63 / Non-inverting gain =2, 4, 8, 16, 32, 64 with VINM0”, 01000 to 01101
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

Figure 175. PGA mode, non-inverting gain setting (x2/x4/x8/x16/x32/x64) or inverting gain setting (x-1/x-3/x-7/x-15/x-31/x-63)



1. *opamp_out can be redirected internally to an ADC channel by setting OPAINTOEN bit. In this case, the I/O on which is mapped the OPAMPx_VOUT is free and can be used for another purpose.*

MSv48043V1

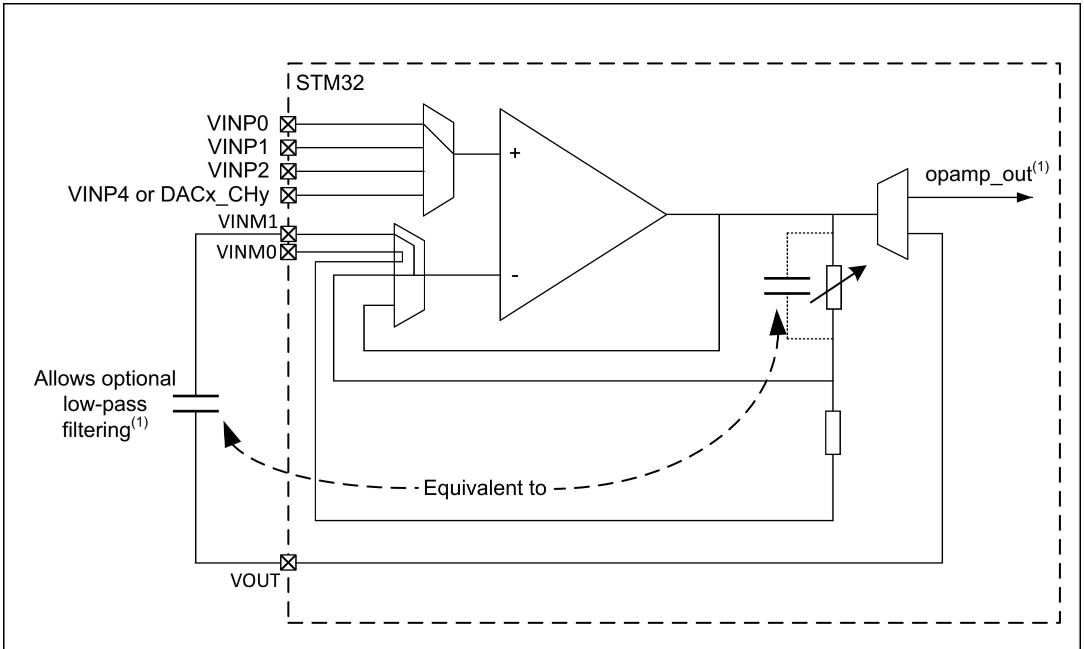
Programmable gain amplifier, non-inverting with external bias or inverting mode with filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal with bias voltage for non-inverting mode or inverting mode with filtering

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “Inverting gain = -1, -3, -7, -15, -31, -63 / Non-inverting gain = 2, 4, 8, 16, 32, 64 with VINM0, VINM1 pin for filtering”, 11000 to 11101
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

  Any external connection on VINM1 can be used in parallel with the internal PGA, for example a capacitor can be connected between opamp_out and VINM1 for filtering purpose (see datasheet for the value of resistors used in the PGA resistor network).

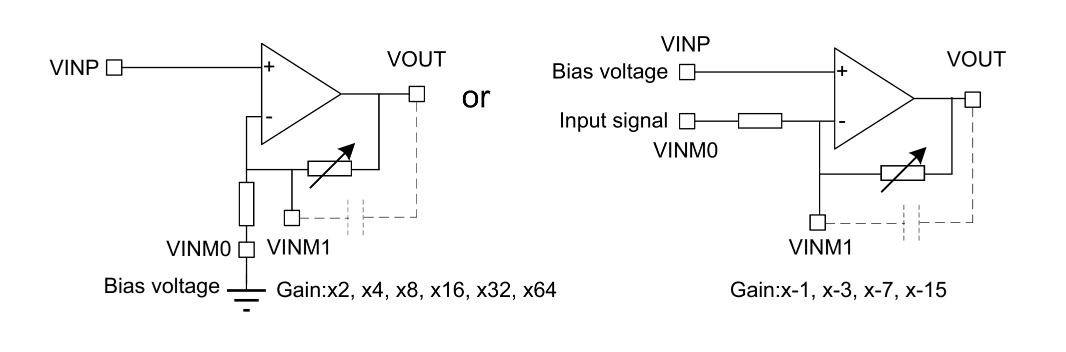
Figure 176. PGA mode, non-inverting gain setting (x2/x4/x8/x16/x32/x64) or inverting gain setting (x-1/x-3/x-7/x-15/x-31/x-63) with filtering



1. opamp_out can be redirected internally to an ADC channel by setting OPAINTOEN bit. In this case, the I/O on which is mapped the OPAMPx_VOUT is free and can be used for another purpose.

MSv48045V1

Figure 177. Example configuration



MSv48044V1

25.3.6 OPAMP PGA gain

When the OPAMP is configured in PGA mode, the gain can be programmed to x2, x4, x8, x16, x32, x64 in non-inverting configuration and x-1, x-3, x-7, x-15, x-31, x-63 for inverting configuration.

When the OPAMP is configured in non-inverting mode, the gain solely depends on internal resistive divider. When it is configured as inverting mode, Gain factor is defined not only the on chip feedback resistor but also the signal source output impedance. If signal source output impedance is not negligible compare to the input feedback resistance of PGA, it creates the gain error.

25.3.7 Calibration

The OPAMP offset value is minimized using a trimming circuitry. At startup, the trimming values are initialized with the preset ‘factory’ trimming value. Each operational amplifier can also be trimmed by the user if the OPAMP is used in conditions different from the factory trimming conditions.

Each operational amplifier can be trimmed by the user. Specific registers allow to have different trimming values for normal mode and for high-speed mode.

The aim of the calibration is to cancel as much as possible the OPAMP inputs offset voltage. The calibration circuitry allows to reduce the input offset voltage to less than +/-3 mV within stable voltage and temperature conditions.

For each operational amplifier and two trimming values (TRIMOFFSETP and TRIMOFFSETN in OPAMPx_CSR register) need to be trimmed, one for N differential pair and one for P differential pair.

The user is able to switch from ‘factory’ values to ‘user’ trimmed values using the USERTRIM bit in the OPAMPx_CSR register. This bit is reset at startup and so the ‘factory’ value are applied by default to the OPAMP option registers.

The offset trimming TRIMOFFSETP and TRIMOFFSETN bits are typically configured after the calibration operation is initialized by setting bit CALON to 1. When CALON = 1 the inputs of the operational amplifier are disconnected from the I/Os.

- Setting CALSEL to 01 initializes the offset calibration for the P differential pair (low voltage reference used).
- Resetting CALSEL to 11 initializes the offset calibration for the N differential pair (high voltage reference used).

When CALON = 1, the bit CALOUT reflects the influence of the trimming value selected by CALSEL and OPAHSM. The software must increment the TRIMOFFSETN bits in the OPAMP control register from 0x00 to the first value that causes the CALOUT bit to change from 1 to 0 in the OPAMP register. If the CALOUT bit is reset, the offset is calibrated correctly and the corresponding trimming value must be stored. The CALOUT flag needs up to 1 ms after the trimming value is changed to become steady (see $t_{OFFTRIM}$max delay specification in the electrical characteristics section of the datasheet).

Note: The closer the trimming value is to the optimum trimming value, the longer it takes to stabilize (with a maximum stabilization time remaining below 1 ms in any case).

Table 201. Operating modes and calibration

Mode	Control bits				Output	
	OPAEN	**OPAHSM**	**CALON**	**CALSEL**	**V_{OUT}**	**CALOUT flag**
Normal operating mode	1	0	0	X	analog	0
High-speed mode	1	1	0	X	analog	0
Power down	0	X	X	X	Z	0
Offset cal N	1	X	1	11	analog	X
Offset cal P diff	1	X	1	01	analog	X

Calibration procedure

Here are the steps to perform a full calibration of either one of the operational amplifiers:

1. Set the OPAEN bit in OPAMPx_CSR to 1 to enable the operational amplifier.
2. Set the USERTRIM bit in the OPAMPx_CSR register to 1.
3. Choose a calibration mode (refer to *Table 201: Operating modes and calibration*). The steps 3 to 4 have to be repeated four times. For the first iteration select Normal mode and N differential pair. This calibration mode correspond to OPAHSM = 0 and CALSEL = 11 in the OPAMPx_CSR register.
4. Increment TRIMOFFSETN[4:0] in OPAMPx_OTR starting from 0b00000 until CALOUT changes to 0 in OPAMPx_CSR.

Note: Between the write to the TRIMOFFSETP and TRIMOFFSETN bits and the read of the CALOUT value, make sure to wait for the $t_{OFFTRIM}$max delay specified in the electrical characteristics section of the datasheet, to get the correct CALOUT value.

The commutation means that the is correctly compensated and that the corresponding trim code must be saved in the TRIMOFFSETP and TRIMOFFSETN bits.

Repeat steps 3 to 4 for:

- Normal mode and P differential pair, CALSEL = 01
- High-speed mode and N differential pair
- High-speed mode and P differential pair

If a mode is not used, it is not necessary to perform the corresponding calibration.

All operational amplifier can be calibrated at the same time.

Note: During the whole calibration phase the external connection of the operational amplifier output must not pull up or down currents higher than 500 µA.

If the OPAMP output is internally connected to an ADC channel and disconnected from the output pin (OPAINTOEN = 1 in the OPAMPx_CSR register), the offset trimming procedure differs from the case where the OPAMP output is connected to the output pin (OPAINTOEN = 0). The calibration procedure is the similar as above but the CALOUT bit change detection cannot be used as indicated in step 4. Instead, the ADC output data must be used as indicator to detect the OPAMP output change: a change of CALOUT from 1 to 0 corresponds to the change of ADC output data from values close to the maximum ADC output to values close to the minimum ADC output (the ADC works as a comparator connected to the OPAMP output). Another solution is to perform the calibration with OPAINTOEN = 0, and then change OPAINTOEN to 1. In this case, the OPAMP output GPIO toggles during the calibration and care must be taken that there is no conflict on this GPIO.

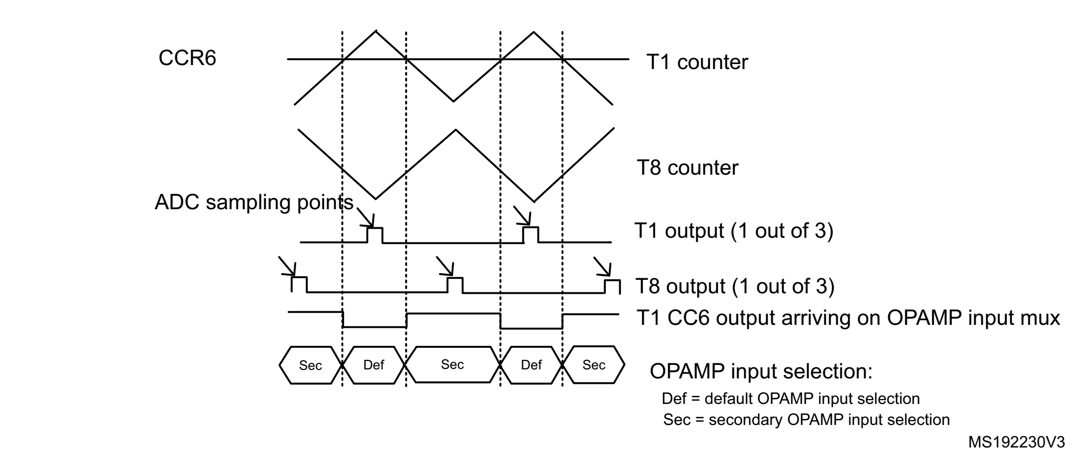
25.3.8 Timer controlled Multiplexer mode

The selection of the OPAMP inverting and non inverting inputs can be done automatically. In this case, the switch from one input to another is done automatically. This automatic switch is triggered by the TIM1 CC6 or TIM8 CC6 or TIM20 CC6 output arriving on the OPAMP input multiplexers.

This is useful for dual motor control with a need to measure the currents on the 3 phases simultaneously on a first motor and then on the second motor.

The automatic switch is enabled by setting the TxCMEN bit, x = 1,8,20, in the OPAMP switch control register. The inverting and non inverting inputs selection is performed using the VPS_SEL and VMS_SEL bit fields in the OPAMP timer controlled mode register. If the TxCMEN bit is cleared, the selection is done using the VP_SEL and VM_SEL bit fields in the OPAMP control/status register.

Figure 178. Timer controlled Multiplexer mode



25.4 OPAMP low-power modes

Table 202. Effect of low-power modes on the OPAMP

Mode	Description
Sleep	No effect.
Low-power run	No effect.
Low-power sleep	No effect.
Stop 0 / Stop 1	No effect, OPAMP registers content is kept.
Standby	The OPAMP registers are powered down and must be re-initialized after exiting Standby or Shutdown mode.
Shutdown	

25.5 OPAMP registers

25.5.1 OPAMP1 control/status register (OPAMP1_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	CAL OUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN		
rw	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
PGA_GAIN		CALSEL		CALON	Res.	Res.	OPA INTOEN	OPA HSM	VM_SEL		USER TRIM.	VP_SEL		FORCE _VP	OPAEN
rw	rw	rw	rw	rw			rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP1_CSR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP1_CSR register as read-only.

0: OPAMP1_CSR is read-write

1: OPAMP1_CSR is read-only

Bit 30 **CALOUT:** Operational amplifier calibration output

This bit shows the digital value of OPAMP output and is the calibration output status during calibration offset mode (Calibration is successful when CALOUT switches from 1 to 0.)

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **TRIMOFFSETN [4:0]**: Trim for NMOS differential pairs

Bits 23:19 **TRIMOFFSETP [4:0]**: Trim for PMOS differential pairs

At reset these bits are loaded by the factory trimming value. They can be modified only when USER_TRIM =1

Bits 18:14 **PGA_GAIN:** Operational amplifier Programmable amplifier gain value

00000: Non inverting internal gain =2
00001: Non inverting internal gain =4
00010: Non inverting internal gain =8
00011: Non inverting internal gain =16
00100: Non inverting internal gain =32
00101: Non inverting internal gain =64
00110: Not used
00111: Not used
01000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias
01001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias
01010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias
01011: Inverting gain = -15/ non inverting gain =16 VINM0 pin for input or bias
01100: Inverting gain = -31/ non inverting gain =32 VINM0 pin for input or bias
01101: Inverting gain = -63/ non inverting gain =64 VINM0 pin for input or bias
01110: Not used
01111: Not used
10000: Non inverting gain =2 with filtering on VINM0 pin
10001: Non inverting gain =4 with filtering on VINM0 pin
10010: Non inverting gain =8 with filtering on VINM0 pin
10011: Non inverting gain =16 with filtering on VINM0 pin
10100: Non inverting gain =32 with filtering on VINM0 pin
10101: Non inverting gain =64 with filtering on VINM0 pin
10110: Not used
10111: Not used
11000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias, VINM1 pin for filtering
11001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias, VINM1 pin for filtering
11010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias, VINM1 pin for filtering
11011: Inverting gain = -15/ non inverting gain =16 with VINM0 pin for input or bias, VINM1 pin for filtering
11100: Inverting gain = -31/ non inverting gain =32 with VINM0 pin for input or bias, VINM1 pin for filtering
11101: Inverting gain = -63/ non inverting gain =64 with VINM0 pin for input or bias, VINM1 pin for filtering
11110: Not used
11111: Not used

Bits 13:12 **CALSEL:** Calibration selection

It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00: 0.033*VDDA applied on OPAMP inputs
01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)
10: 0.5*VDDA applied on OPAMP inputs
11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enabled

0: Normal mode
1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAINTOEN:** Operational amplifier internal output enable.

0: The OPAMP output is connected to the output pin

1: The OPAMP output is connected internally to an ADC channel and disconnected from the output pin

Bit 7 **OPAHSM:** Operational amplifier high-speed mode

The operational amplifier must be disable to change this configuration.

0: Operational amplifier in normal mode

1: Operational amplifier in high-speed mode

Bits 6:5 **VM_SEL:** Inverting input selection

00: VINM0 pin connected to OPAMP VINM input

01: VINM1 pin connected to OPAMP VINM input

10: Feedback resistor is connected to OPAMP VINM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting

11: opamp_out connected to OPAMP VINM input (Follower mode)

Bit 4 **USERTRIM**: User trimming enable

This bit allows to switch from 'factory' AOP offset trimmed values to 'user' AOP offset trimmed values

This bit is active for both mode normal and high-power.

0: 'factory' trim code used

1: 'user' trim code used

Bits 3:2 **VP_SEL:** Non inverted input selection

00: VINP0 pin connected to OPAMP1 VINP input

01: VINP1 pin connected to OPAMP1 VINP input

10: VINP2 pin connected to OPAMP1 VINP input

11: DAC3_CH1 connected to OPAMP1 VINP input

Bit 1 **FORCE_VP:** Force internal reference on OPAMP VINP input (reserved for test)

0: Normal operating mode. Non-inverting input connected to inputs.

1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN:** Operational amplifier Enable

0: Operational amplifier disabled

1: Operational amplifier enabled

25.5.2 OPAMP2 control/status register (OPAMP2_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	CAL OUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN		
rw	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
PGA_GAIN		CALSEL		CALON	Res.	Res.	OPA INTOEN	OPA HSM	VM_SEL		USER TRIM.	VP_SEL		FORCE _VP	OPAEN
rw	rw	rw	rw	rw			rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP2_CSR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP2_CSR register as read-only.

0: OPAMP2_CSR is read-write
1: OPAMP2_CSR is read-only

Bit 30 **CALOUT:** Operational amplifier calibration output

This bit shows the digital value of OPAMP output and is the calibration output status during calibration offset mode (Calibration is successful when CALOUT switches from 1 to 0.)

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **TRIMOFFSETN [4:0]**: Trim for NMOS differential pairs

Bits 23:19 **TRIMOFFSETP [4:0]**: Trim for PMOS differential pairs

At reset these bits are loaded by the factory trimming value. They can be modified only when USER_TRIM =1

Bits 18:14 **PGA_GAIN:** Operational amplifier Programmable amplifier gain value

00000: Non inverting internal gain =2
00001: Non inverting internal gain =4
00010: Non inverting internal gain =8
00011: Non inverting internal gain =16
00100: Non inverting internal gain =32
00101: Non inverting internal gain =64
00110: Not used
00111: Not used
01000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias
01001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias
01010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias
01011: Inverting gain = -15/ non inverting gain =16 VINM0 pin for input or bias
01100: Inverting gain = -31/ non inverting gain =32 VINM0 pin for input or bias
01101: Inverting gain = -63/ non inverting gain =64 VINM0 pin for input or bias
01110: Not used
01111: Not used
10000: Non inverting gain =2 with filtering on VINM0 pin
10001: Non inverting gain =4 with filtering on VINM0 pin
10010: Non inverting gain =8 with filtering on VINM0 pin
10011: Non inverting gain =16 with filtering on VINM0 pin
10100: Non inverting gain =32 with filtering on VINM0 pin
10101: Non inverting gain =64 with filtering on VINM0 pin
10110: Not used
10111: Not used
11000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias, VINM1 pin for filtering
11001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias, VINM1 pin for filtering
11010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias, VINM1 pin for filtering
11011: Inverting gain = -15/ non inverting gain =16 with VINM0 pin for input or bias, VINM1 pin for filtering
11100: Inverting gain = -31/ non inverting gain =32 with VINM0 pin for input or bias, VINM1 pin for filtering
11101: Inverting gain = -63/ non inverting gain =64 with VINM0 pin for input or bias, VINM1 pin for filtering
11110: Not used
11111: Not used

Bits 13:12 **CALSEL:** Calibration selection

It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00: 0.033*VDDA applied on OPAMP inputs
01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)
10: 0.5*VDDA applied on OPAMP inputs
11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enabled

0: Normal mode
1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAINTOEN:** Operational amplifier internal output enable.

0: The OPAMP output is connected to the output pin

1: The OPAMP output is connected internally to an ADC channel and disconnected from the output pin

Bit 7 **OPAHSM:** Operational amplifier high-speed mode

The operational amplifier must be disable to change this configuration.

0: Operational amplifier in normal mode

1: Operational amplifier in high-speed mode

Bits 6:5 **VM_SEL:** Inverting input selection

00: VINM0 pin connected to OPAMP VINM input

01: VINM1 pin connected to OPAMP VINM input

10: Feedback resistor is connected to OPAMP VINM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting

11: opamp_out connected to OPAMP VINM input (Follower mode)

Bit 4 **USERTRIM**: User trimming enable

This bit allows to switch from ‘factory’ AOP offset trimmed values to ‘user’ AOP offset trimmed values

This bit is active for both mode normal and high-power.

0: ‘factory’ trim code used

1: ‘user’ trim code used

Bits 3:2 **VP_SEL:** Non inverted input selection

00: VINP0 pin connected to OPAMP2 VINP input

01: VINP1 pin connected to OPAMP2 VINP input

10: VINP2 pin connected to OPAMP2 VINP input

11: VINP3 pin connected to OPAMP2 VINP input

Bit 1 **FORCE_VP:** Force internal reference on VP (reserved for test)

0: Normal operating mode. Non-inverting input connected to inputs.

1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN:** Operational amplifier Enable

0: Operational amplifier disabled

1: Operational amplifier enabled

25.5.3 OPAMP3 control/status register (OPAMP3_CSR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	CAL OUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN		
rw	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
PGA_GAIN		CALSEL		CALON	Res.	Res.	OPA INTOEN	OPA HSM	VM_SEL		USER TRIM.	VP_SEL		FORCE _VP	OPAEN
rw	rw	rw	rw	rw			rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP3_CSR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP3_CSR register as read-only.

0: OPAMP3_CSR is read-write

1: OPAMP3_CSR is read-only

Bit 30 **CALOUT:** Operational amplifier calibration output

This bit shows the digital value of OPAMP output and is the calibration output status during calibration offset mode (Calibration is successful when CALOUT switches from 1 to 0.)

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **TRIMOFFSETN [4:0]**: Trim for NMOS differential pairs

Bits 23:19 **TRIMOFFSETP [4:0]**: Trim for PMOS differential pairs

At reset these bits are loaded by the factory trimming value. They can be modified only when USER_TRIM =1

Bits 18:14 **PGA_GAIN:** Operational amplifier Programmable amplifier gain value

00000: Non inverting internal gain =2
00001: Non inverting internal gain =4
00010: Non inverting internal gain =8
00011: Non inverting internal gain =16
00100: Non inverting internal gain =32
00101: Non inverting internal gain =64
00110: Not used
00111: Not used
01000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias
01001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias
01010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias
01011: Inverting gain = -15/ non inverting gain =16 VINM0 pin for input or bias
01100: Inverting gain = -31/ non inverting gain =32 VINM0 pin for input or bias
01101: Inverting gain = -63/ non inverting gain =64 VINM0 pin for input or bias
01110: Not used
01111: Not used
10000: Non inverting gain =2 with filtering on VINM0 pin
10001: Non inverting gain =4 with filtering on VINM0 pin
10010: Non inverting gain =8 with filtering on VINM0 pin
10011: Non inverting gain =16 with filtering on VINM0 pin
10100: Non inverting gain =32 with filtering on VINM0 pin
10101: Non inverting gain =64 with filtering on VINM0 pin
10110: Not used
10111: Not used
11000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias, VINM1 pin for filtering
11001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias, VINM1 pin for filtering
11010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias, VINM1 pin for filtering
11011: Inverting gain = -15/ non inverting gain =16 with VINM0 pin for input or bias, VINM1 pin for filtering
11100: Inverting gain = -31/ non inverting gain =32 with VINM0 pin for input or bias, VINM1 pin for filtering
11101: Inverting gain = -63/ non inverting gain =64 with VINM0 pin for input or bias, VINM1 pin for filtering
11110: Not used
11111: Not used

Bits 13:12 **CALSEL:** Calibration selection

It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00: 0.033*VDDA applied on OPAMP inputs
01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)
10: 0.5*VDDA applied on OPAMP inputs
11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enabled

0: Normal mode
1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAINTOEN:** Operational amplifier internal output enable.

0: The OPAMP output is connected to the output pin

1: The OPAMP output is connected internally to an ADC channel and disconnected from the output pin

Bit 7 **OPAHSM:** Operational amplifier high-speed mode

The operational amplifier must be disable to change this configuration.

0: Operational amplifier in normal mode

1: Operational amplifier in high-speed mode

Bits 6:5 **VM_SEL:** Inverting input selection

00: VINM0 pin connected to OPAMP VINM input

01: VINM1 pin connected to OPAMP VINM input

10: Feedback resistor is connected to OPAMP VINM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting

11: opamp_out connected to OPAMP VINM input (Follower mode)

Bit 4 **USERTRIM**: User trimming enable

This bit allows to switch from ‘factory’ AOP offset trimmed values to ‘user’ AOP offset trimmed values

This bit is active for both mode normal and high-power.

0: ‘factory’ trim code used

1: ‘user’ trim code used

Bits 3:2 **VP_SEL:** Non inverted input selection

00: VINP0 pin connected to OPAMP3 VINP input

01: VINP1 pin connected to OPAMP3 VINP input

10: VINP2 pin connected to OPAMP3 VINP input

11: DAC3_CH2 connected to OPAMP3 VINP input

Bit 1 **FORCE_VP:** Force internal reference on VP (reserved for test)

0: Normal operating mode. Non-inverting input connected to inputs.

1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN:** Operational amplifier Enable

0: Operational amplifier disabled

1: Operational amplifier enabled

25.5.4 OPAMP4 control/status register (OPAMP4_CSR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	CAL OUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN		
rw	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
PGA_GAIN		CALSEL		CALON	Res.	Res.	OPA INTOEN	OPA HSM	VM_SEL		USER TRIM.	VP_SEL		FORCE _VP	OPAEN
rw	rw	rw	rw	rw			rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP4_CSR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP4_CSR register as read-only.

0: OPAMP4_CSR is read-write

1: OPAMP4_CSR is read-only

Bit 30 **CALOUT:** Operational amplifier calibration output

This bit shows the digital value of OPAMP output and is the calibration output status during calibration offset mode (Calibration is successful when CALOUT switches from 1 to 0.)

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **TRIMOFFSETN [4:0]**: Trim for NMOS differential pairs

Bits 23:19 **TRIMOFFSETP [4:0]**: Trim for PMOS differential pairs

At reset these bits are loaded by the factory trimming value. They can be modified only when USER_TRIM =1

Bits 18:14 **PGA_GAIN:** Operational amplifier Programmable amplifier gain value

00000: Non inverting internal gain =2
00001: Non inverting internal gain =4
00010: Non inverting internal gain =8
00011: Non inverting internal gain =16
00100: Non inverting internal gain =32
00101: Non inverting internal gain =64
00110: Not used
00111: Not used
01000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias
01001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias
01010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias
01011: Inverting gain = -15/ non inverting gain =16 VINM0 pin for input or bias
01100: Inverting gain = -31/ non inverting gain =32 VINM0 pin for input or bias
01101: Inverting gain = -63/ non inverting gain =64 VINM0 pin for input or bias
01110: Not used
01111: Not used
10000: Non inverting gain =2 with filtering on VINM0 pin
10001: Non inverting gain =4 with filtering on VINM0 pin
10010: Non inverting gain =8 with filtering on VINM0 pin
10011: Non inverting gain =16 with filtering on VINM0 pin
10100: Non inverting gain =32 with filtering on VINM0 pin
10101: Non inverting gain =64 with filtering on VINM0 pin
10110: Not used
10111: Not used
11000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias, VINM1 pin for filtering
11001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias, VINM1 pin for filtering
11010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias, VINM1 pin for filtering
11011: Inverting gain = -15/ non inverting gain =16 with VINM0 pin for input or bias, VINM1 pin for filtering
11100: Inverting gain = -31/ non inverting gain =32 with VINM0 pin for input or bias, VINM1 pin for filtering
11101: Inverting gain = -63/ non inverting gain =64 with VINM0 pin for input or bias, VINM1 pin for filtering
11110: Not used
11111: Not used

Bits 13:12 **CALSEL:** Calibration selection

It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00: 0.033*VDDA applied on OPAMP inputs
01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)
10: 0.5*VDDA applied on OPAMP inputs
11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enabled

0: Normal mode
1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAINTOEN:** Operational amplifier internal output enable.

0: The OPAMP output is connected to the output pin

1: The OPAMP output is connected internally to an ADC channel and disconnected from the output pin

Bit 7 **OPAHSM:** Operational amplifier high-speed mode

The operational amplifier must be disable to change this configuration.

0: Operational amplifier in normal mode

1: Operational amplifier in high-speed mode

Bits 6:5 **VM_SEL:** Inverting input selection

00: VINM0 pin connected to OPAMP VINM input

01: VINM1 pin connected to OPAMP VINM input

10: Feedback resistor is connected to OPAMP VINM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting

11: opamp_out connected to OPAMP VINM input (Follower mode)

Bit 4 **USERTRIM**: User trimming enable

This bit allows to switch from ‘factory’ AOP offset trimmed values to ‘user’ AOP offset trimmed values

This bit is active for both mode normal and high-power.

0: ‘factory’ trim code used

1: ‘user’ trim code used

Bits 3:2 **VP_SEL:** Non inverted input selection

00: VINP0 pin connected to OPAMP4 VINP input

01: VINP1 pin connected to OPAMP4 VINP input

10: VINP2 pin connected to OPAMP4 VINP input

11: DAC4_CH1 connected to OPAMP4 VINP input

Bit 1 **FORCE_VP:** Force internal reference on VP (reserved for test)

0: Normal operating mode. Non-inverting input connected to inputs.

1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN:** Operational amplifier Enable

0: Operational amplifier disabled

1: Operational amplifier enabled

25.5.5 OPAMP5 control/status register (OPAMP5_CSR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	CAL OUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN		
rw	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
PGA_GAIN		CALSEL		CALON	Res.	Res.	OPA INTOEN	OPA HSM	VM_SEL		USER TRIM.	VP_SEL		FORCE _VP	OPAEN
rw	rw	rw	rw	rw			rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP5_CSR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP5_CSR register as read-only.

0: OPAMP5_CSR is read-write

1: OPAMP5_CSR is read-only

Bit 30 **CALOUT:** Operational amplifier calibration output

This bit shows the digital value of OPAMP output and is the calibration output status during calibration offset mode (Calibration is successful when CALOUT switches from 1 to 0.)

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **TRIMOFFSETN [4:0]**: Trim for NMOS differential pairs

Bits 23:19 **TRIMOFFSETP [4:0]**: Trim for PMOS differential pairs

At reset these bits are loaded by the factory trimming value. They can be modified only when USER_TRIM =1

Bits 18:14 **PGA_GAIN:** Operational amplifier Programmable amplifier gain value

00000: Non inverting internal gain =2
00001: Non inverting internal gain =4
00010: Non inverting internal gain =8
00011: Non inverting internal gain =16
00100: Non inverting internal gain =32
00101: Non inverting internal gain =64
00110: Not used
00111: Not used
01000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias
01001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias
01010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias
01011: Inverting gain = -15/ non inverting gain =16 VINM0 pin for input or bias
01100: Inverting gain = -31/ non inverting gain =32 VINM0 pin for input or bias
01101: Inverting gain = -63/ non inverting gain =64 VINM0 pin for input or bias
01110: Not used
01111: Not used
10000: Non inverting gain =2 with filtering on VINM0 pin
10001: Non inverting gain =4 with filtering on VINM0 pin
10010: Non inverting gain =8 with filtering on VINM0 pin
10011: Non inverting gain =16 with filtering on VINM0 pin
10100: Non inverting gain =32 with filtering on VINM0 pin
10101: Non inverting gain =64 with filtering on VINM0 pin
10110: Not used
10111: Not used
11000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias, VINM1 pin for filtering
11001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias, VINM1 pin for filtering
11010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias, VINM1 pin for filtering
11011: Inverting gain = -15/ non inverting gain =16 with VINM0 pin for input or bias, VINM1 pin for filtering
11100: Inverting gain = -31/ non inverting gain =32 with VINM0 pin for input or bias, VINM1 pin for filtering
11101: Inverting gain = -63/ non inverting gain =64 with VINM0 pin for input or bias, VINM1 pin for filtering
11110: Not used
11111: Not used

Bits 13:12 **CALSEL:** Calibration selection

It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00: 0.033*VDDA applied on OPAMP inputs
01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)
10: 0.5*VDDA applied on OPAMP inputs
11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enabled

0: Normal mode
1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAINTOEN:** Operational amplifier internal output enable.

0: The OPAMP output is connected to the output pin

1: The OPAMP output is connected internally to an ADC channel and disconnected from the output pin

Bit 7 **OPAHSM:** Operational amplifier high-speed mode

The operational amplifier must be disable to change this configuration.

0: Operational amplifier in normal mode

1: Operational amplifier in high-speed mode

Bits 6:5 **VM_SEL:** Inverting input selection

00: VINM0 pin connected to OPAMP VINM input

01: VINM1 pin connected to OPAMP VINM input

10: Feedback resistor is connected to OPAMP VINM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting

11: opamp_out connected to OPAMP VINM input (Follower mode)

Bit 4 **USERTRIM**: User trimming enable

This bit allows to switch from ‘factory’ AOP offset trimmed values to ‘user’ AOP offset trimmed values

This bit is active for both mode normal and high-power.

0: ‘factory’ trim code used

1: ‘user’ trim code used

Bits 3:2 **VP_SEL:** Non inverted input selection

00: VINP0 pin connected to OPAMP5 VINP input

01: VINP1 pin connected to OPAMP5 VINP input

10: VINP2 pin connected to OPAMP5 VINP input

11: DAC4_CH2 connected to OPAMP5 VINP input

Bit 1 **FORCE_VP:** Force internal reference on VP (reserved for test)

0: Normal operating mode. Non-inverting input connected to inputs.

1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN:** Operational amplifier Enable

0: Operational amplifier disabled

1: Operational amplifier enabled

25.5.6 OPAMP6 control/status register (OPAMP6_CSR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	CAL OUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN		
rw	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**
PGA_GAIN		CALSEL		CALON	Res.	Res.	OPA INTOEN	OPA HSM	VM_SEL		USER TRIM.	VP_SEL		FORCE _VP	OPAEN
rw	rw	rw	rw	rw			rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP6_CSR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP6_CSR register as read-only.

0: OPAMP6_CSR is read-write

1: OPAMP6_CSR is read-only

Bit 30 **CALOUT:** Operational amplifier calibration output

This bit shows the digital value of OPAMP output and is the calibration output status during calibration offset mode (Calibration is successful when CALOUT switches from 1 to 0.)

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **TRIMOFFSETN [4:0]**: Trim for NMOS differential pairs

Bits 23:19 **TRIMOFFSETP [4:0]**: Trim for PMOS differential pairs

At reset these bits are loaded by the factory trimming value. They can be modified only when USER_TRIM =1

Bits 18:14 **PGA_GAIN:** Operational amplifier Programmable amplifier gain value

00000: Non inverting internal gain =2
00001: Non inverting internal gain =4
00010: Non inverting internal gain =8
00011: Non inverting internal gain =16
00100: Non inverting internal gain =32
00101: Non inverting internal gain =64
00110: Not used
00111: Not used
01000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias
01001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias
01010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias
01011: Inverting gain = -15/ non inverting gain =16 VINM0 pin for input or bias
01100: Inverting gain = -31/ non inverting gain =32 VINM0 pin for input or bias
01101: Inverting gain = -63/ non inverting gain =64 VINM0 pin for input or bias
01110: Not used
01111: Not used
10000: Non inverting gain =2 with filtering on VINM0 pin
10001: Non inverting gain =4 with filtering on VINM0 pin
10010: Non inverting gain =8 with filtering on VINM0 pin
10011: Non inverting gain =16 with filtering on VINM0 pin
10100: Non inverting gain =32 with filtering on VINM0 pin
10101: Non inverting gain =64 with filtering on VINM0 pin
10110: Not used
10111: Not used
11000: Inverting gain = -1 / non inverting gain =2 with VINM0 pin for input or bias, VINM1 pin for filtering
11001: Inverting gain = -3 / non inverting gain =4 with VINM0 pin for input or bias, VINM1 pin for filtering
11010: Inverting gain = -7 / non inverting gain =8 with VINM0 pin for input or bias, VINM1 pin for filtering
11011: Inverting gain = -15/ non inverting gain =16 with VINM0 pin for input or bias, VINM1 pin for filtering
11100: Inverting gain = -31/ non inverting gain =32 with VINM0 pin for input or bias, VINM1 pin for filtering
11101: Inverting gain = -63/ non inverting gain =64 with VINM0 pin for input or bias, VINM1 pin for filtering
11110: Not used
11111: Not used

Bits 13:12 **CALSEL:** Calibration selection

It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00: 0.033*VDDA applied on OPAMP inputs
01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)
10: 0.5*VDDA applied on OPAMP inputs
11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enabled

0: Normal mode
1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAINTOEN:** Operational amplifier internal output enable.

0: The OPAMP output is connected to the output pin

1: The OPAMP output is connected internally to an ADC channel and disconnected from the output pin

Bit 7 **OPAHSM:** Operational amplifier high-speed mode

The operational amplifier must be disable to change this configuration.

0: Operational amplifier in normal mode

1: Operational amplifier in high-speed mode

Bits 6:5 **VM_SEL:** Inverting input selection

00: VINM0 pin connected to OPAMP VINM input

01: VINM1 pin connected to OPAMP VINM input

10: Feedback resistor is connected to OPAMP VINM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting

11: opamp_out connected to OPAMP VINM input (Follower mode)

Bit 4 **USERTRIM**: User trimming enable

This bit allows to switch from ‘factory’ AOP offset trimmed values to ‘user’ AOP offset trimmed values

This bit is active for both mode normal and high-power.

0: ‘factory’ trim code used

1: ‘user’ trim code used

Bits 3:2 **VP_SEL:** Non inverted input selection

00: VINP0 pin connected to OPAMP6 VINP input

01: VINP1 pin connected to OPAMP6 VINP input

10: VINP2 pin connected to OPAMP6 VINP input

11: DAC3_CH1 connected to OPAMP6 VINP input

Bit 1 **FORCE_VP:** Force internal reference on VP (reserved for test)

0: Normal operating mode. Non-inverting input connected to inputs.

1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN:** Operational amplifier Enable

0: Operational amplifier disabled

1: Operational amplifier enabled

25.5.7 OPAMP1 timer controlled mode register (OPAMPx_TCMR) (x = 1...6)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.														
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.										T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
										rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP1_TCMR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP1_TCMR register as read-only.

0: OPAMP1_TCMR is read-write

1: OPAMP1_TCMR is read-only

Bits 30:6 Reserved, must be kept at reset value.

Bit 5 **T20CM_EN**: TIM20 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM20 CC6 output arriving on the OPAMPx input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 4 **T8CM_EN**: TIM8 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM8 CC6 output arriving on the OPAMPx input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 3 **T1CM_EN**: TIM1 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMPx input multiplexers.

0: Automatic switch is disabled.
1: Automatic switch is enabled.

Bits 2:1 **VPS_SEL**: OPAMP1 Non inverting input secondary selection.

These bits are set and cleared by software. They are used to select the OPAMPx non inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

00: VINP0 pin connected to OPAMP1 VINP input
01: VINP1 pin connected to OPAMP1 VINP input
10: VINP2 pin connected to OPAMP1 VINP input
11: DAC3_CH1 connected to OPAMP1 VINP input

Bits 0 **VMS_SEL**: OPAMP1 inverting input secondary selection

This bit is set and cleared by software. It is used to select the OPAMPx inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

When standalone mode is used (i.e. VM_SEL = “00” or “01”):

0: Input from VINM0
1: Input from VINM1

When PGA (VM_SEL = “10”) or Follower mode (VM_SEL = “11”) is used:

0: Resistor feedback output selected (PGA mode)
1: V_{OUT} selected as input minus (follower mode)

25.5.8 OPAMP2 timer controlled mode register (OPAMPx_TCMR) (x = 1...6)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.														
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.										T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
										rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP2_TCMR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP2_TCMR register as read-only.

0: OPAMP2_TCMR is read-write

1: OPAMP2_TCMR is read-only

Bits 30:6 Reserved, must be kept at reset value.

Bit 5 **T20CM_EN**: TIM20 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM20 CC6 output arriving on the OPAMP2 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 4 **T8CM_EN**: TIM8 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM8 CC6 output arriving on the OPAMP2 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 3 **T1CM_EN**: TIM1 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMP2 input multiplexers.

0: Automatic switch is disabled.
1: Automatic switch is enabled.

Bits 2:1 **VPS_SEL**: OPAMP2 Non inverting input secondary selection.

These bits are set and cleared by software. They are used to select the OPAMP2 non inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

00: VINP0 pin connected to OPAMP2 VINP input
01: VINP1 pin connected to OPAMP2 VINP input
10: VINP2 pin connected to OPAMP2 VINP input
11: VINP3 pin connected to OPAMP2 VINP input

Bits 0 **VMS_SEL**: OPAMP2 inverting input secondary selection

This bit is set and cleared by software. It is used to select the OPAMPx inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

When standalone mode is used (i.e. VM_SEL = “00” or “01”):

0: Input from VINM0
1: Input from VINM1

When PGA (VM_SEL = “10”) or Follower mode (VM_SEL = “11”) is used:

0: Resistor feedback output selected (PGA mode)
1: V_{OUT} selected as input minus (follower mode)

25.5.9 OPAMP3 timer controlled mode register (OPAMPx_TCMR) (x = 1...6)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.														
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.										T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
										rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP3_TCMR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP3_TCMR register as read-only.

0: OPAMP3_TCMR is read-write

1: OPAMP3_TCMR is read-only

Bits 30:6 Reserved, must be kept at reset value.

Bit 5 **T20CM_EN**: TIM20 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM20 CC6 output arriving on the OPAMP3 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 4 **T8CM_EN**: TIM8 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM8 CC6 output arriving on the OPAMP3 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 3 **T1CM_EN**: TIM1 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMP3 input multiplexers.

0: Automatic switch is disabled.
1: Automatic switch is enabled.

Bits 2:1 **VPS_SEL**: OPAMP3 Non inverting input secondary selection.

These bits are set and cleared by software. They are used to select the OPAMP3 non inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

00: VINP0 pin connected to OPAMP3 VINP input
01: VINP1 pin connected to OPAMP3 VINP input
10: VINP2 pin connected to OPAMP3 VINP input
11: DAC3_CH2 connected to OPAMP3 VINP input

Bits 0 **VMS_SEL**: OPAMP3 inverting input secondary selection

This bit is set and cleared by software. It is used to select the OPAMPx inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

When standalone mode is used (i.e. VM_SEL = “00” or “01”):

0: Input from VINM0
1: Input from VINM1

When PGA (VM_SEL = “10”) or Follower mode (VM_SEL = “11”) is used:

0: Resistor feedback output selected (PGA mode)
1: V_{OUT} selected as input minus (follower mode)

25.5.10 OPAMP4 timer controlled mode register (OPAMPx_TCMR) (x = 1...6)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.														
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.										T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
										rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP4_TCMR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP4_TCMR register as read-only.

0: OPAMP4_TCMR is read-write

1: OPAMP4_TCMR is read-only

Bits 30:6 Reserved, must be kept at reset value.

Bit 5 **T20CM_EN**: TIM20 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM20 CC6 output arriving on the OPAMP4 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 4 **T8CM_EN**: TIM8 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM8 CC6 output arriving on the OPAMP4 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 3 **T1CM_EN**: TIM1 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMP4 input multiplexers.

0: Automatic switch is disabled.
1: Automatic switch is enabled.

Bits 2:1 **VPS_SEL**: OPAMP4 Non inverting input secondary selection.

These bits are set and cleared by software. They are used to select the OPAMP4 non inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

00: VINP0 pin connected to OPAMP4 VINP input
01: VINP1 pin connected to OPAMP4 VINP input
10: VINP2 pin connected to OPAMP4 VINP input
11: DAC4_CH1 connected to OPAMP4 VINP input

Bits 0 **VMS_SEL**: OPAMP4 inverting input secondary selection

This bit is set and cleared by software. It is used to select the OPAMP4 inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1).

When standalone mode is used (i.e. VM_SEL = “00” or “01”):

0: Input from VINM0
1: Input from VINM1

When PGA (VM_SEL = “10”) or Follower mode (VM_SEL = “11”) is used:

0: Resistor feedback output selected (PGA mode)
1: V_{OUT} selected as input minus (follower mode)

25.5.11 OPAMP5 timer controlled mode register (OPAMPx_TCMR) (x = 1...6)

Address offset: 0x28

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.										T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
										rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP5_TCMR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP5_TCMR register as read-only.

0: OPAMP5_TCMR is read-write

1: OPAMP5_TCMR is read-only

Bits 30:6 Reserved, must be kept at reset value.

Bit 5 **T20CM_EN**: TIM20 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM20 CC6 output arriving on the OPAMP5 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 4 **T8CM_EN**: TIM8 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM8 CC6 output arriving on the OPAMP5 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 3 **T1CM_EN**: TIM1 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMP5 input multiplexers.

0: Automatic switch is disabled.
1: Automatic switch is enabled.

Bits 2:1 **VPS_SEL**: OPAMP5 Non inverting input secondary selection.

These bits are set and cleared by software. They are used to select the OPAMP5 non inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

00: VINP0 pin connected to OPAMP5 VINP input
01: VINP1 pin connected to OPAMP5 VINP input
10: VINP2 pin connected to OPAMP5 VINP input
11: DAC4_CH2 connected to OPAMP5 VINP input

Bits 0 **VMS_SEL**: OPAMP5 inverting input secondary selection

This bit is set and cleared by software. It is used to select the OPAMP5 inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

When standalone mode is used (i.e. VM_SEL = “00” or “01”):

0: Input from VINM0
1: Input from VINM1

When PGA (VM_SEL = “10”) or Follower mode (VM_SEL = “11”) is used:

0: Resistor feedback output selected (PGA mode)
1: V_{OUT} selected as input minus (follower mode)

25.5.12 OPAMP6 timer controlled mode register (OPAMPx_TCMR) (x = 1...6)

Address offset: 0x2C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.										T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
										rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP6_TCMR lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP6_TCMR register as read-only.

0: OPAMP6_TCMR is read-write

1: OPAMP6_TCMR is read-only

Bits 30:6 Reserved, must be kept at reset value.

Bit 5 **T20CM_EN**: TIM20 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM20 CC6 output arriving on the OPAMP6 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 4 **T8CM_EN**: TIM8 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM8 CC6 output arriving on the OPAMP6 input multiplexers.

0: Automatic switch is disabled.

1: Automatic switch is enabled.

Bit 3 **T1CM_EN**: TIM1 controlled mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMP6 input multiplexers.

0: Automatic switch is disabled.
1: Automatic switch is enabled.

Bits 2:1 **VPS_SEL**: OPAMP6 Non inverting input secondary selection.

These bits are set and cleared by software. They are used to select the OPAMP6 non inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

00: VINP0 pin connected to OPAMP6 VINP input
01: VINP1 pin connected to OPAMP6 VINP input
10: VINP2 pin connected to OPAMP6 VINP input
11: DAC3_CH1 connected to OPAMP6 VINP input

Bits 0 **VMS_SEL**: OPAMP6 inverting input secondary selection

This bit is set and cleared by software. It is used to select the OPAMP6 inverting input when the controlled mux mode is enabled (T1CM_EN = 1 or T8CM_EN = 1 or T20CM_EN = 1)

When standalone mode is used (i.e. VM_SEL = “00” or “01”):

0: Input from VINM0
1: Input from VINM1

When PGA (VM_SEL = “10”) or Follower mode (VM_SEL = “11”) is used:

0: Resistor feedback output selected (PGA mode)
1: V_{OUT} selected as input minus (follower mode)

25.5.13 OPAMP register map

Table 203. OPAMP 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	OPAMP1_CSR	LOCK	CALOUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN					CALSEL		CALON	Res.	Res.	OPAINTOEN	OPAHSM	VM_SEL		USERTRIM	VP_SEL		FORCE_VP	OPAEN
	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
0x04	OPAMP2_CSR	LOCK	CALOUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN					CALSEL		CALON	Res.	Res.	OPAINTOEN	OPAHSM	VM_SEL		USERTRIM	VP_SEL		FORCE_VP	OPAEN
	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	OPAMP3_CSR	LOCK	CALOUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN					CALSEL		CALON	Res.	Res.	OPAINTOEN	OPAHSM	VM_SEL		USERTRIM	VP_SEL		FORCE_VP	OPAEN
	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	OPAMP4_CSR	LOCK	CALOUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN					CALSEL		CALON	Res.	Res.	OPAINTOEN	OPAHSM	VM_SEL		USERTRIM	VP_SEL		FORCE_VP	OPAEN
	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
0x10	OPAMP5_CSR	LOCK	CALOUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN					CALSEL		CALON	Res.	Res.	OPAINTOEN	OPAHSM	VM_SEL		USERTRIM	VP_SEL		FORCE_VP	OPAEN
	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	OPAMP6_CSR	LOCK	CALOUT	Res.	TRIMOFFSETN					TRIMOFFSETP					PGA_GAIN					CALSEL		CALON	Res.	Res.	OPAINTOEN	OPAHSM	VM_SEL		USERTRIM	VP_SEL		FORCE_VP	OPAEN
	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	OPAMP1_ TCMR	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
	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	OPAMP2_ TCMR	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
	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	OPAMP3_ TCMR	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
	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
0x24	OPAMP4_ TCMR	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
	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. OPAMP 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	**OPAMP5_ TCMR**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
	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	**OPAMP6_ TCMR**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T20CM_EN	T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
	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 2.2 on page 81* for the register boundary addresses.

26 True random number generator (RNG)

26.1 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 can be used to construct a NIST compliant Deterministic Random Bit Generator (DRBG), acting as a live entropy source.

The RNG true random number generator has been tested using German BSI statistical tests of AIS-31 (T0 to T8).

26.2 RNG main features

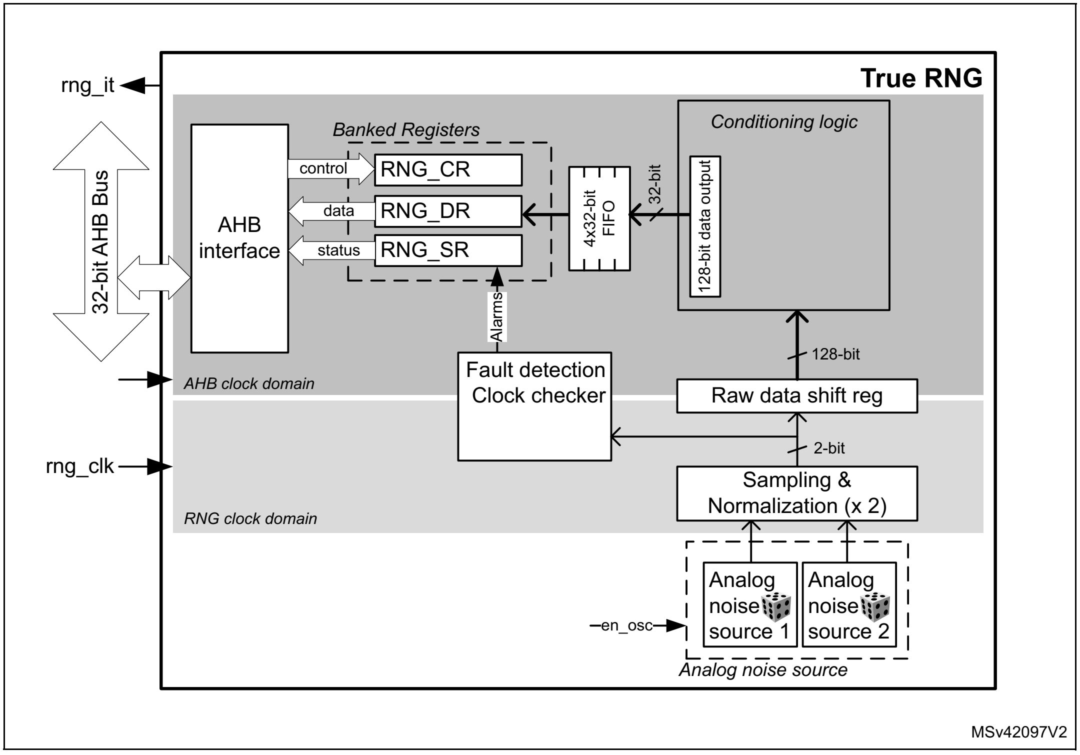
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source processed by a high quality conditioning stage.
- It produces four 32-bit random samples every 16x $\frac{f_{AHB}}{f_{RNG}}$ AHB clock cycles, if value is higher than 213 cycles (213 cycles otherwise).
- It allows embedded continuous basic health tests with associated error management
  - Includes too low sampling clock detection and repetition count tests.
- It can be disabled to reduce power consumption.
- 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).

26.3 RNG functional description

26.3.1 RNG block diagram

Figure 179 shows the RNG block diagram.

Figure 179. RNG block diagram



26.3.2 RNG internal signals

Table 204 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 204. RNG internal input/output signals

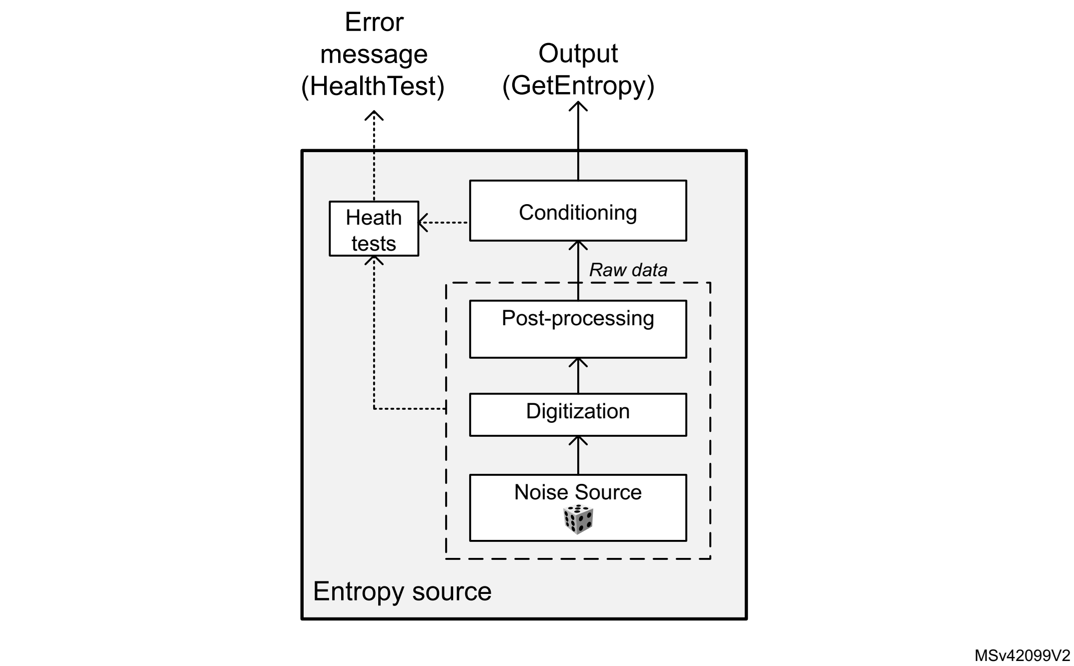
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk

26.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals. Within its boundary the RNG implements the entropy source model pictured on *Figure 180*.

It includes an analog noise source, a digitization stage with post-processing, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

Figure 180. 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. It is composed of:

- Two analog noise sources, 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 26.3.8: RNG low-power usage*.
- A sampling stage of these outputs clocked by a dedicated clock input (**rng_clk**), delivering a 2-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (**rng_hclk**).

Note: In Section 26.6: RNG entropy source validation recommended RNG clock frequencies are given.

Post processing

The sample values obtained from a true random noise source consist of 2-bit bitstrings. Because this noise source output is biased, the RNG implements a post-processing component that reduces that bias to a tolerable level.

More specifically, for each of the two noise source bits the RNG takes half of the bits from the sampled noise source, and half of the bits from inverted sampled noise source. Thus, if the source generates more ‘1’ than ‘0’ (or the opposite), it is filtered

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).

Also note that post-processing computations are triggered when at least 32 bits of raw data are received and when output FIFO needs a refill. Thus the RNG output entropy is maximum when the RNG 128-bit FIFO is emptied by application after 64 RNG clock cycles.

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in *Section 26.5: RNG processing time*.

The conditioning component is clocked by the faster AHB clock.

Output buffer

A data output buffer can store up to four 32-bit words which 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 213 AHB clock cycles later.

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.

1. Continuous health tests, running indefinitely on the output of the noise source
   - Repetition count test, flagging an error when:
   a) One of the noise source has provided more than 64 consecutive bits at a constant value (“0” or “1”), or more than 32 consecutive occurrence of two bits patterns (“01” or “10”)
   b) Both noise sources have delivered more than 32 consecutive bits at a constant value (“0” or “1”), or more than 16 consecutive occurrence of two bits patterns (“01” or “10”)
2. Vendor specific continuous test
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle is smaller than AHB clock cycle divided by 32.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 26.3.7: Error management*.

Note: An interrupt can be generated when an error is detected.

26.3.4 RNG initialization

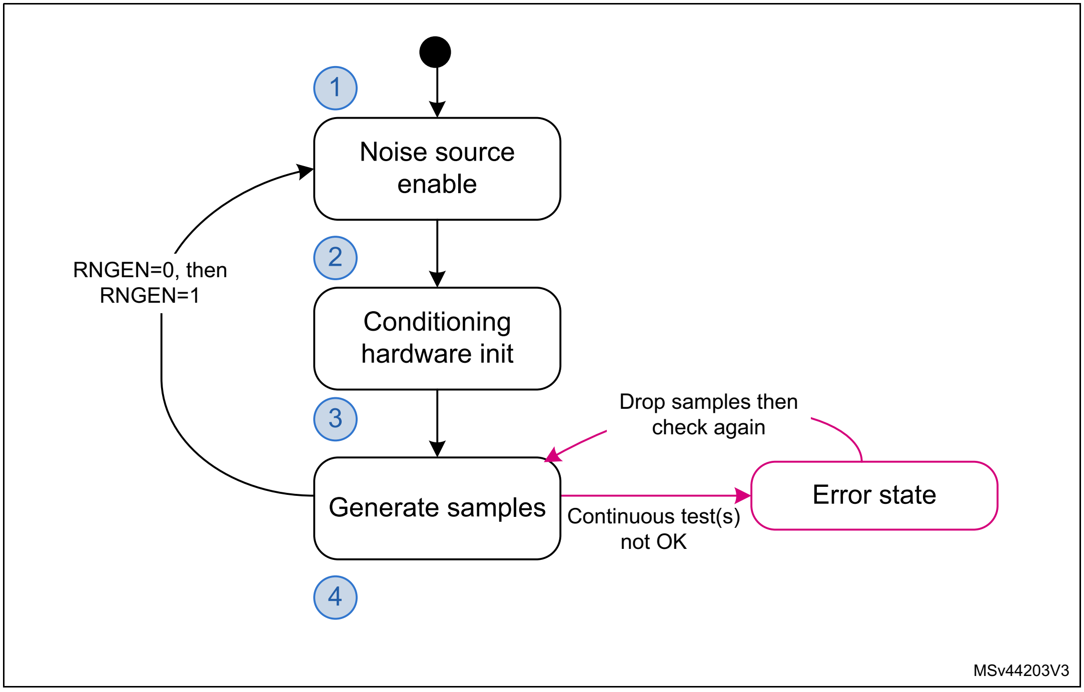
The RNG simplified state machine is pictured on *Figure 181*

After enabling the RNG (RNGEN=1 in RNG_CR) the following chain of events occurs:

1. The analog noise source is enabled, and logic immediately starts sampling the analog output, filling 128-bit conditioning shift register
2. The conditioning logic is enabled and post-processing context is initialized using two 128 noise source bits.
3. The conditioning stage internal input data buffer is filled again with 128-bit and one conditioning round is performed. The output buffer is then filled with post processing result.
4. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 26.5: RNG processing time*.

Figure 181. RNG initialization overview



26.3.5 RNG operation

Normal operations

To run the RNG using interrupts the following steps are recommended:

1. Enable the interrupts by setting the IE bit in the RNG_CR register. At the same time enable the RNG by setting the 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 should 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 26.3.7* must be used.

To run the RNG in polling mode following steps are recommended:

1. Enable the random number generation 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 should be set to 0)
   - A random number is ready (the DRDY bit should 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 26.3.7* 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).

Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 26.3.8: RNG low-power usage*.

Software post-processing

If a NIST approved DRBG with 128 bits of security strength is required an approved random generator software must be built around the RNG true random number generator.

Built-in health check functions are described in *Section 26.3.3: Random number generation*.

26.3.6 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 is used for noise source sampling. Recommended clock configurations are detailed in *Section 26.6: RNG entropy source validation*.

*Note: When the CED bit in the RNG_CR register is set to “0”, the RNG clock frequency **should 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 26.3.1: RNG block diagram* for details (AHB and RNG clock domains).

26.3.7 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 should check that the RNG clock is configured correctly (see *Section 26.3.6: 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, i.e. 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. If the error was detected during the initialization phase the whole initialization sequence is automatically restarted by the RNG.

The following sequence must be used to fully recover from a seed error after the RNG initialization:

1. Clear the SEIS bit by writing it to "0".
2. Read out 12 words from the RNG_DR register, and discard each of them in order to clean the pipeline.
3. Confirm that SEIS is still cleared. Random number generation is back to normal.

26.3.8 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 the RNG must be re-enabled by the application until at least 32 new bits are collected from the noise source and a complete conditioning round is done. It corresponds to 16 RNG clock cycles to sample new bits, and 216 AHB clock cycles to run a conditioning round.

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 (i.e. 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".

26.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 26.3.7: Error management*
- Clock error, see *Section 26.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 205*.

Table 205. 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
	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.

26.5 RNG processing time

The conditioning stage can produce four 32-bit random numbers every 16x $\frac{f_{AHB}}{f_{RNG}}$ clock cycles, if the value is higher than 213 cycles (213 cycles otherwise).
More time is needed for the first set of random numbers after the device exits reset (see *Section 26.3.4: RNG initialization*). Indeed, after enabling the RNG for the first time, random data is first available after either:

- 128 RNG clock cycles + 426 AHB cycles, if $f_{AHB} < f_{threshold}$
- 192 RNG clock cycles + 213 AHB cycles, if $f_{AHB} \geq f_{threshold}$

With $f_{threshold} = (213 \text{ x } f_{RNG})/ 64$

26.6 RNG entropy source validation

26.6.1 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). The results can be provided on demand or the customer can reproduce the tests.

26.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk= 48 MHz (CED bit = ’0’ in RNG_CR register) and rng_clk = 400 kHz (CED bit = ‘1’ in RNG_CR register).

26.6.3 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.

Contact STMicroelectronics if above samples need to be retrieved for your product.

26.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

26.7.1 RNG control register (RNG_CR)

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.	Res.	Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
										rw		rw	rw		

Bits 31:6 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, i.e. to enable or disable CED the RNG must be disabled.

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.

26.7.2 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. 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 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:

- One of the noise source has provided more than 64 consecutive bits at a constant value (“0” or “1”), or more than 32 consecutive occurrence of two bit patterns (“01” or “10”)

Both noise sources have delivered more than 32 consecutive bits at a constant value (“0” or “1”), or more than 16 consecutive occurrence of two bit patterns (“01” or “10”)

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.

26.7.3 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. After being read this register delivers a new random value after 216 periods of AHB clock if the output FIFO is empty.

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.

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).

26.7.4 RNG register map

Table 206 gives the RNG register map and reset values.

Table 206. 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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value																											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

Refer to *Section 2.2* for the register boundary addresses.

27 High-resolution timer (HRTIM)

27.1 Introduction

The high-resolution timer can generate up to 12 digital signals with highly accurate timings. It is primarily intended to drive power conversion systems such as switch mode power supplies or lighting systems, but can be of general purpose usage, whenever a very fine timing resolution is expected.

Its modular architecture allows to generate either independent or coupled waveforms. The wave-shape is defined by self-contained timings (using counters and compare units) and a broad range of external events, such as analog or digital feedbacks and synchronization signals. This allows to produce a large variety of control signal (PWM, phase-shifted, constant Ton,...) and address most of conversion topologies.

For control and monitoring purposes, the timer has also timing measure capabilities and links to built-in ADC and DAC converters. Last, it features light-load management mode and is able to handle various fault schemes for safe shut-down purposes.

27.2 Main features

- High-resolution timing units
  - 184 ps resolution, compensated against voltage and temperature variations
  - High-resolution available on all outputs, possibility to adjust duty-cycle, frequency and pulse width in triggered one-pulse mode
  - 6 16-bit timing units (each one with an independent counter and 4 compare units)
  - 12 outputs that can be controlled by any timing unit, up to 32 set/reset sources per channel
  - Modular architecture to address either multiple independent converters with 1 or 2 switches or few large multi-switch topologies
- Up to 10 external events, available for any timing unit
  - Programmable polarity and edge sensitivity
  - 5 events with a fast asynchronous mode
  - 5 events with a programmable digital filter
  - Spurious events filtering with blanking and windowing modes
- Multiple links to built-in analog peripherals
  - 4 triggers to ADC converters
  - 3 triggers to DAC converters
  - 3 comparators for analog signal conditioning
- Versatile protection scheme
  - 6 fault inputs can be combined and associated to any timing unit
  - Programmable polarity, edge sensitivity, and programmable digital filter
  - Dedicated delayed protections for resonant converters
- Multiple HRTIM instances can be synchronized with external synchronization inputs/outputs
- Versatile output stage
  - High-resolution deadtime insertion (down to 735 ps)
  - Programmable output polarity
  - Chopper mode
- Burst mode controller to handle light-load operation synchronously on multiple converters
- 8 interrupt vectors, each one with up to 14 sources
- 7 DMA requests with up to 14 sources, with a burst mode for multiple registers update

27.3 Functional description

27.3.1 General description

The HRTIM can be partitioned into several sub entities:

- The master timer
- The timing units (timer A to timer F)
- The output stage
- The burst mode controller
- An external event and fault signal conditioning logic that is shared by all timers
- The system interface

The master timer is based on a 16-bit up counter. It can set/reset any of the 12 outputs via 4 compare units and it provides synchronization signals to the 6 timer units. Its main purpose is to have the timer units controlled by a unique source. An interleaved buck converter is a typical application example where the master timer manages the phase-shifts between the multiple units.

The timer units are working either independently or coupled with the other timers including the master timer. Each timer contains the controls for two outputs. The outputs set/reset events are triggered either by the timing units compare registers or by events coming from the master timer, from the other timers or from external events.

The output stage has several duties

- Addition of deadtime when the 2 outputs are configured in complementary PWM mode
- Addition of a carrier frequency on top of the modulating signal
- Management of fault events, by asynchronously asserting the outputs to a predefined safe level

The burst mode controller can take over the control of one or multiple timers in case of light-load operation. The burst length and period can be programmed, as well as the idle state of the outputs.

The external event and fault signal conditioning logic includes:

- The input selection MUXes (for instance for selecting a digital input or an on-chip source for a given external event channel)
- Polarity and edge-sensitivity programming
- Digital filtering (for 5 channels out of 12)

The system interface allows the HRTIM to interact with the rest of the MCU:

- Interrupt requests to the CPU
- DMA controller for automatic accesses to/from the memories, including an HRTIM specific burst mode
- Triggers for the ADC and DAC converters

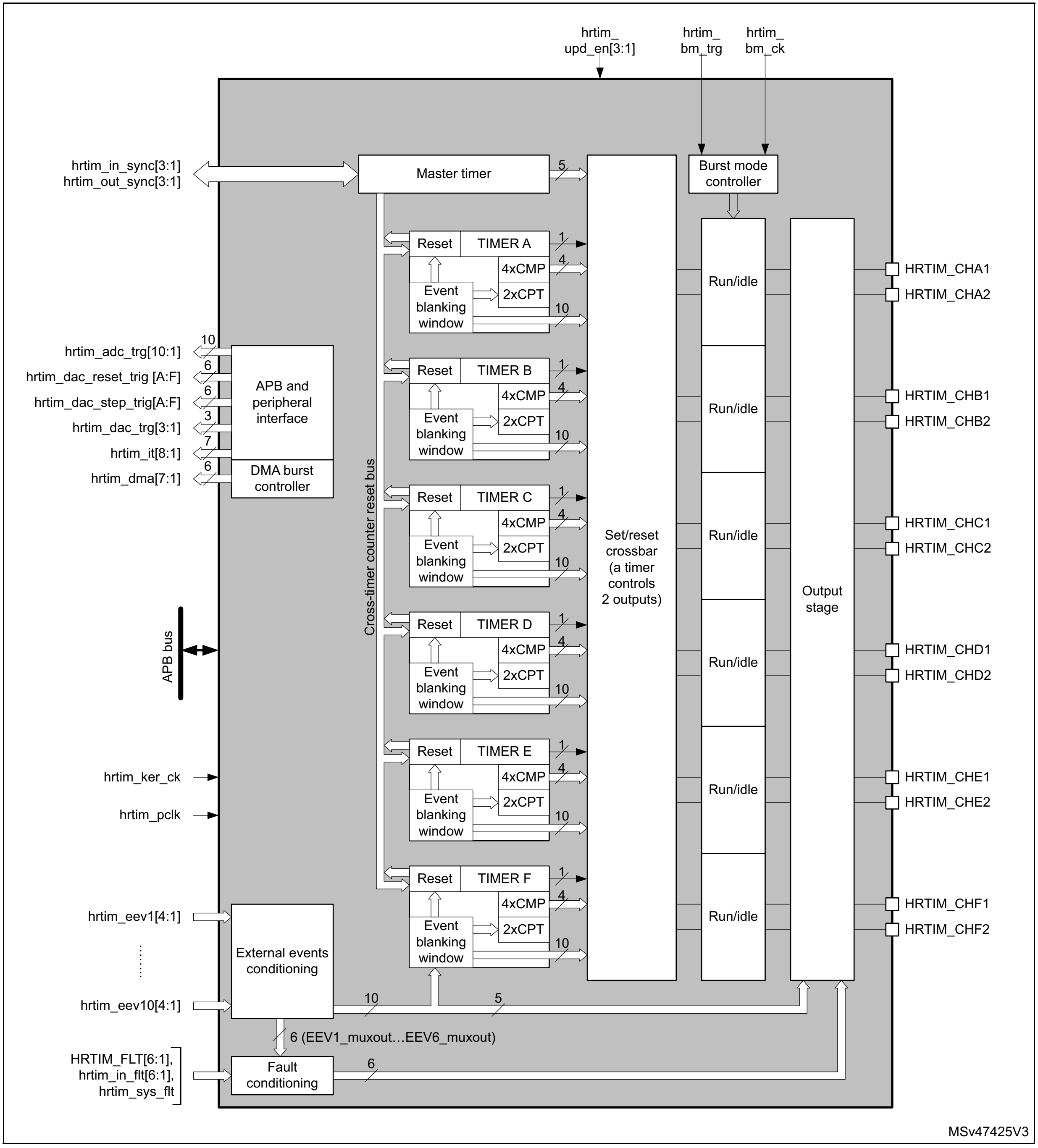
The HRTIM registers are split into 8 groups:

- Master timer registers
- Timer A to timer F registers
- Common registers for features shared by all timer units

Note: As a writing convention, references to the 6 timing units in the text and in registers are generalized using the "x" letter, where x can be any value from A to F.

The block diagram of the timer is shown in *Figure 182*.

Figure 182. High-resolution timer overview



27.3.2 HRTIM pins and internal signals

The tables in this section summarize the HRTIM inputs and outputs, both on-chip and off-chip.

Table 207. HRTIM inputs/outputs summary

Signal name	Signal type	Description
HRTIM_CHA1, HRTIM_CHA2, HRTIM_CHB1, HRTIM_CHB2, HRTIM_CHC1, HRTIM_CHC2, HRTIM_CHD1, HRTIM_CHD2, HRTIM_CHE1, HRTIM_CHE2, HRTIM_CHF1, HRTIM_CHF2	Outputs	Main HRTIM timer outputs. They can be coupled by pairs (HRTIM_CHx1 & HRTIM_CHx2) with deadtime insertion or work independently.
hrtim_in_flt1[4:1] hrtim_in_flt2[4:1] hrtim_in_flt3[4:1] hrtim_in_flt4[4:1] hrtim_in_flt5[4:1] hrtim_in_flt6[4:1]	Digital input	Fault inputs: immediately disable the HRTIM outputs when asserted (12 on-chip inputs and 6 off-chip HRTIM_FLTx inputs).
hrtim_sys_flt	Digital input	System fault gathering MCU internal fault events (clock security system, SRAM parity error, Cortex®-M4 with FPU lockup (HardFault), PVD output.
hrtim_in_sync[3:1]	Digital Input	Synchronization inputs to synchronize the whole HRTIM with other internal or external timer resources: hrtim_in_sync1: Reserved hrtim_in_sync2: the source is the TIM1_TRGO output hrtim_in_sync3: the source is HRTIM_SCIN input pins
hrtim_out_sync[2:1]	Digital output	The purpose of this output is to cascade or synchronize several HRTIM instances, either on-chip or off-chip: hrtim_out_sync1: Reserved hrtim_out_sync2: the destination is an off-chip HRTIM or peripheral (via HRTIM_SCOUT output pins)

Table 207. HRTIM inputs/outputs summary (continued)

Signal name	Signal type	Description
hrtim_eev1[4:1]	Digital input	External events. Each of the 10 events can be selected among 4 sources, either on-chip (from other built-in peripherals: comparator, ADC analog watchdog, TIMx timers, trigger outputs) or off-chip (HRTIM_EEVx input pins).
hrtim_eev2[4:1]		
hrtim_eev3[4:1]		
hrtim_eev4[4:1]		
hrtim_eev5[4:1]		
hrtim_eev6[4:1]		
hrtim_eev7[4:1]		
hrtim_eev8[4:1]		
hrtim_eev9[4:1]		
hrtim_eev10[4:1]		
hrtim_upd_en[3:1]	Digital input	A pulse on the update enable inputs hrtim_upd_end[3:1] (on-chip interconnect) triggers the transfer from shadow to active registers
hrtim_bm_trg	Digital input	A pulse on this input triggers a burst mode entry. This input is connected to the TIM7_TRGO output.
hrtim_bm_ck[4:1]	Digital input	Burst mode clock (on-chip interconnect)
hrtim_adc_trg[10:1]	Digital output	ADC start of conversion triggers. The hrtim_adc_trg1 .. hrtim_adc_trg10 outputs are here-after referred to as ADC trigger 1 to ADC trigger 10.
hrtim_dac_trg[3:1]	Digital output	DAC conversion update triggers
hrtim_dac_reset_trg[6:1] hrtim_dac_step_trg[6:1]	Digital output	Dual channel DAC triggers
hrtim_it[8:1]	Digital output	Interrupt requests
hrtim_dma[7:1]	Digital output	DMA requests
hrtim_pclk	-	APB clock
hrtim_ker_ck	-	HRTIM kernel clock (APB2 clock)

Table 208. External events mapping and associated features

External event channel	Fast mode	Digital filter	Balanc-ed fault timer A,B,C	Balanc-ed fault timer D,E,F	Source (EExSRC[1:0])				Comparator and input sources available per package	
					Src1 (00)	Src 2 (01)	Src3 (10)	Src4 (11)	48-pin	64-pin and 100-pin
hrtim_eev 1[4:1]	Yes	-	-	-	PC12	COMP2	TIM1_ TRGO	ADC1_ AWD1	Comp	Comp & input
hrtim_eev 2[4:1]	Yes	-	-	-	PC11	COMP4	TIM2_ TRGO	ADC1_ AWD2	Comp	Comp & input
hrtim_eev 3[4:1]	Yes	-	-	-	PB7	COMP6	TIM3_ TRGO	ADC1_ AWD3	Comp & input	Comp & input
hrtim_eev 4[4:1]	Yes	-	-	-	PB6	COMP1	COMP5	ADC2_ AWD1	Comp & input	Comp & input
hrtim_eev 5[4:1]	Yes	-	-	-	PB9	COMP3	COMP7	ADC2_ AWD2	Comp & input	Comp & input
hrtim_eev 6[4:1]	-	Yes	Yes	-	PB5	COMP2	COMP1	ADC2_ AWD3	Comp & input	Comp & input
hrtim_eev 7[4:1]	-	Yes	Yes	-	PB4	COMP4	TIM7_ TRGO	ADC3_ AWD1	Comp & input	Comp & input
hrtim_eev 8[4:1]	-	Yes	-	Yes	PB8	COMP6	COMP3	ADC4_ AWD1	Comp & input	Comp & input
hrtim_eev 9[4:1]	-	Yes	-	Yes	PB3	COMP5	TIM15_ TRGO	COMP4	Comp & input	Comp & input
hrtim_eev 10[4:1]	-	Yes	-	-	PC5/ PC6	COMP7	TIM6_ TRGO	ADC5_ AWD1	-	Comp & input

Table 209. Update enable inputs and sources

hrtim_upd_en[3:1]	Update source
hrtim_upd_en1	TIM16_OC
hrtim_upd_en2	TIM17_OC
hrtim_upd_en3	TIM6_TRGO

Table 210. Burst mode clock sources from general purpose timer

hrtim_bm_ck[4:1]	BMCLK[3:0]	Clock source
hrtim_bm_ck1	0110	TIM16 OC
hrtim_bm_ck2	0111	TIM17 OC
hrtim_bm_ck3	1000	TIM7 TRGO
hrtim_bm_ck4	1001	Reserved

Table 211. Fault inputs

Fault channel	External Input FLTxSRC[1:0] = 00	On-chip source FLTxSRC[1:0] = 01	External Input FLTxSRC[1:0] = 10	On-chip source FLTxSRC[1:0] = 11
hrtim_in_flt1[4:1]	PA12	COMP2	EEV1	N/A
hrtim_in_flt2[4:1]	PA15	COMP4	EEV2	N/A
hrtim_in_flt3[4:1]	PB10	COMP6	EEV3	N/A
hrtim_in_flt4[4:1]	PB11	COMP1	EEV4	N/A
hrtim_in_flt5[4:1]	PB0/PC7	COMP3	EEV5	N/A
hrtim_in_flt6[4:1]	PC10	COMP5	EEV6	N/A

Table 212. HRTIM DAC triggers connections

HRTIM DAC triggers	DAC1_CH1 DAC_CH2	DAC2_CH1	DAC3_CH1 DAC3_CH2	DAC4_CH1 DAC4_CH2
hrtim_dac_trg1	Yes	-	-	Yes
hrtim_dac_trig2	-	Yes	-	-
hrtim_dac_trig3	-	-	Yes	-
hrtim_dac_reset_trg1 hrtim_dac_step_trg1	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg2 hrtim_dac_step_trg2	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg3 hrtim_dac_step_trg3	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg4 hrtim_dac_step_trg4	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg5 hrtim_dac_step_trg5	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg6 hrtim_dac_step_trg6	Yes	Yes	Yes	Yes

27.3.3 Clocks

The HRTIM must be supplied by the t_{HRTIM} APB2 clock to offer a full resolution. The t_{HRTIM} clock period is evenly divided into up to 32 intermediate steps using an edge positioning logic. All clocks present in the HRTIM are derived from this reference clock.

Definition of terms

f_{HRTIM}: main HRTIM clock (hrtim_ker_ck). All subsequent clocks are derived and synchronous with this source.

f_{HRCK}: high-resolution equivalent clock. Considering the f_{HRTIM} clock period division by 32, it is equivalent to a frequency of 170 x 32 = 5.44 GHz.

f_{DTG}: deadtime generator clock. For convenience, only the t_{DTG} period (t_{DTG} = $1/f_{DTG}$) is used in this document.

f_{CHPFRQ}: chopper stage clock source.

f_{1STPW}: clock source defining the length of the initial pulse in chopper mode. For convenience, only the t_{1STPW} period (t_{1STPW} = $1/f_{1STPW}$) is used in this document.

f_{BRST}: burst mode controller counter clock.

$f_{SAMPLING}$: clock needed to sample the fault or the external events inputs.

f_{FLTS}: clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter fault events.

f_{EEVS}: clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter external events.

Timer clock and prescaler

Each timer in the HRTIM has its own individual clock prescaler, which allows you to adjust the timer resolution (see *Table 213*).

Table 213. Timer resolution and min. PWM frequency for f_{HRTIM} = 170 MHz

CKPSC[2:0]	Prescal-ing ratio	f_{HRCK} equivalent frequency	Resolution	Min PWM frequency
000	1	170 x 32 MHz = 5.44 GHz	184 ps	83.0 kHz
001	2	170 x 16 MHz = 2.72 GHz	368 ps	41.5 kHz
010	4	170 x 8 MHz = 1.36 GHz	735 ps	20.8 kHz
011	8	170 x 4 MHz = 680 MHz	1.47 ns	10.4 kHz
100	16	170 x 2 MHz = 340 MHz	2.94 ns	5.19 kHz
101	32	170 MHz	5.88 ns	2.59 kHz
110	64	170/2 MHz = 85 MHz	11.76 ns	1.30 kHz
111	128	170/4 MHz = 42.5 MHz	23.53 ns	0.65 kHz

The high-resolution is available for edge positioning, PWM period adjustment and externally triggered pulse duration.

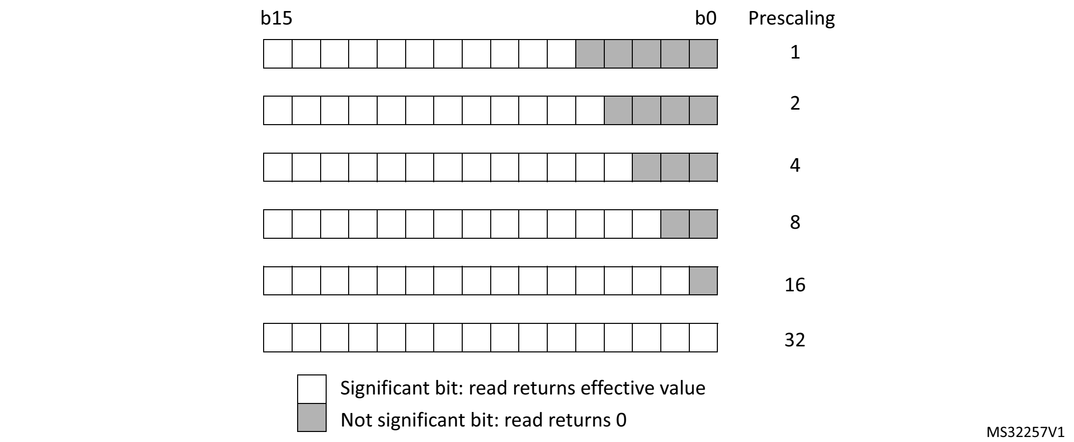
The high-resolution is not available for the following features

- Timer counter read and write accesses
- Capture unit

For clock prescaling ratios below 32 (CKPSC[2:0] < 5), the least significant bits of the counter and capture registers are not significant. The least significant bits cannot be written (counter register only) and return 0 when read.

For instance, if CKPSC[2:0] = 2 (prescaling by 4), writing 0xFFFF into the counter register yields an effective value of 0xFFF8. Conversely, any counter value between 0xFFFF and 0xFFF8 is read as 0xFFF8.

Figure 183. Counter and capture register format vs clock prescaling factor


b15
b0
Prescaling
1
2
4
8
16
32
Significant bit: read returns effective value
Not significant bit: read returns 0
MS32257V1

Initialization

At start-up, it is mandatory to initialize first the prescaler bitfields before writing the compare and period registers. Once the timer is enabled (MCEN or TxCEN bit set in the HRTIM_MCR register), the prescaler cannot be modified.

When multiple timers are enabled, the prescalers are synchronized with the prescaler of the timer that was started first.

Warning: **It is possible to have different prescaling ratios in the master and TIMA..E timers only if the counter and output behavior does not depend on other timers’ information and signals. It is mandatory to configure identical prescaling ratios in these timers when one of the following events is propagated from one timing unit (or master timer) to another: output set/reset event, counter reset event, update event, external event filter or capture triggers. Prescaler factors not equal yield to unpredictable results.**

Deadtime generator clock

The deadtime prescaler is supplied by (f_{HRTIM} x 8) / $2^{(DTPRSC[2:0])}$, programmed with DTPRSC[2:0] bits in the HRTIM_DTxR register.

t_{DTG} ranges from 735 ps to 94.1 ns for f_{HRTIM} = 170 MHz.

Chopper stage clock

The chopper stage clock source f_{CHPFRQ} is derived from f_{HRTIM} with a division factor ranging from 16 to 256, so that 664.06 kHz ≤ f_{CHPFRQ} ≤ 10.625 MHz for f_{HRTIM} = 170 MHz.

t_{1STPW} is the length of the initial pulse in chopper mode, programmed with the STRPW[3:0] bits in the HRTIM_CHPxR register, as follows:

t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.

It uses f_{HRTIM} / 16 as clock source (10.625 MHz for f_{HRTIM}= 170 MHz).

Burst mode prescaler

The burst mode controller counter clock f_{BRST} can be supplied by several sources, among which one is derived from f_{HRTIM}.

In this case, f_{BRST} ranges from f_{HRTIM} to f_{HRTIM} / 32768 (5.188 kHz for f_{HRTIM} = 170 MHz).

Fault input sampling clock

The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{FLTS}.

f_{FLTS} is derived from f_{HRTIM} and ranges from 170 MHz to 21.25 MHz for f_{HRTIM} = 170 MHz.

External event input sampling clock

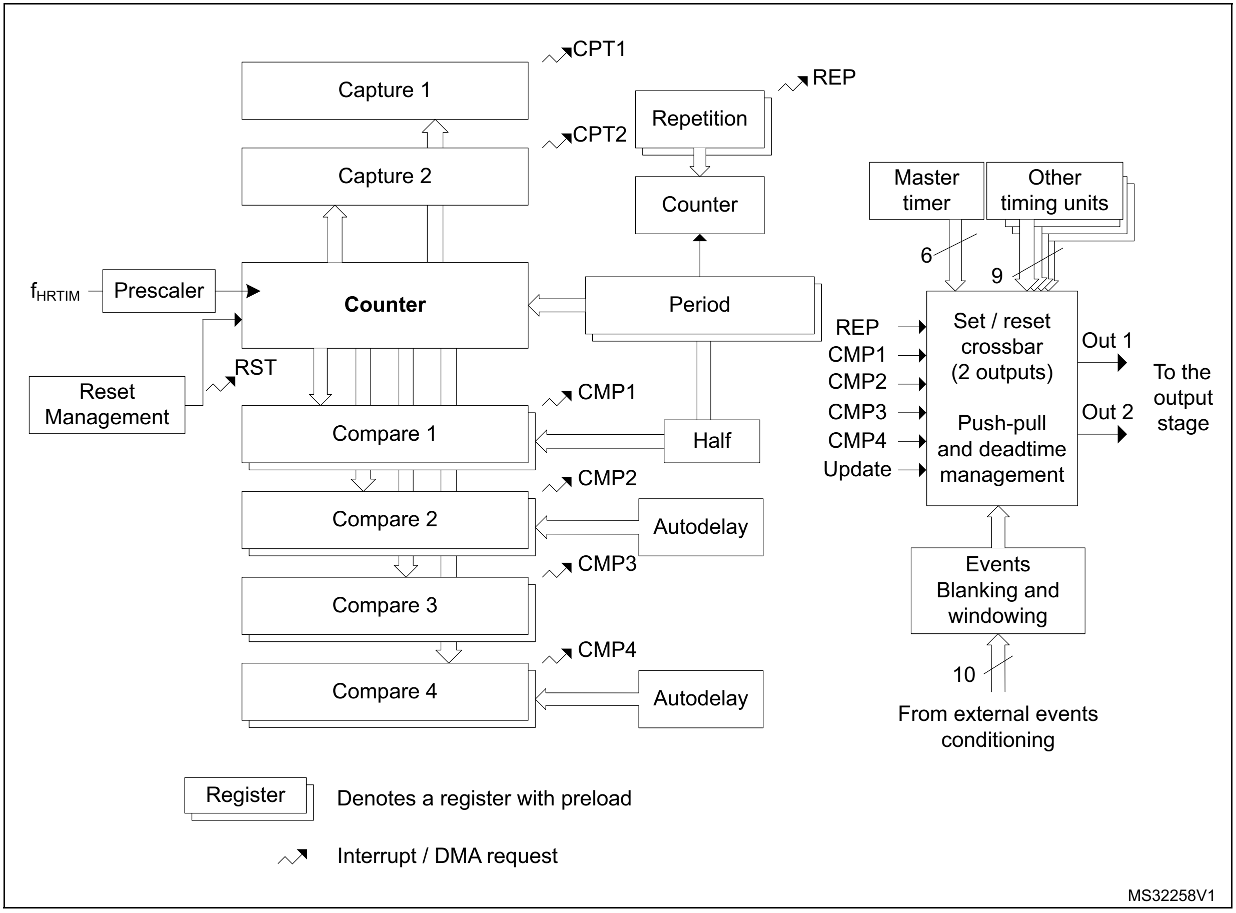
The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{EEVS}.

f_{EEVS} is derived from f_{HRTIM} and ranges from 170 MHz to 21.25 MHz for f_{HRTIM} = 170 MHz.

27.3.4 Timer A..F timing units

The HRTIM embeds 6 identical timing units made of a 16-bit up-counter with an auto-reload mechanism to define the counting period, 4 compare and 2 capture units, as per *Figure 184*. Each unit includes all control features for 2 outputs, so that it operates as a standalone timer.

Figure 184. Timer A..F overview



The period and compare values must be within a lower and an upper limit related to the high-resolution implementation and listed in *Table 214*:

- The minimum value must be greater than or equal to 3 periods of the f_{HRTIM} clock. The value 0x0000 can be written in CMP1 and CMP3 registers only, to skip a PWM pulse. See *Section : Null duty cycle exception case* for details
- The maximum value must be less than or equal to 0xFFFF - 1 periods of the f_{HRTIM} clock.

Table 214. Period and compare registers min and max values

CKPSC[2:0] value	Min[1]	Max
0	0x0060	0xFFDF
1	0x0030	0xFFEF
2	0x0018	0xFFF7
3	0x000C	0xFFFB
4	0x0006	0xFFFD
≥ 5	0x0003	0xFFFD

1. The value 0x0000 can be written in CMP1 and CMP3 registers only, to skip a PWM pulse. See *Section : Null duty cycle exception case* for details.

Note: A compare value greater than the period register value does not generate a compare match event.

Counter operating mode

Timer A..F operate in continuous (free-running) mode or in single-shot manner where counting is started by a reset event, using the CONT bit in the HRTIM_TIMxCR control register. An additional RETRIG bit allows you to select whether the single-shot operation is retriggerable or non-retriggerable. Details of operation are summarized on *Table 215* and on *Figure 185* and *Figure 186*.

Table 215. Timer operating modes

CONT	RETRIG	Operating mode	Start / stop conditions Clocking and event generation
0	0	Single-shot Non-retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A first reset event starts the counting and any subsequent reset is ignored until the counter reaches the PER value. The PER event is then generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
0	1	Single-shot Retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A reset event starts the counting if the counter is stopped, otherwise it clears the counter. When the counter reaches the PER value, the PER event is generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
1	X	Continuous mode	Setting the TxEN bit enables the timer and starts the counter simultaneously. When the counter reaches the PER value, it rolls-over to 0x0000 and resumes counting. The counter can be reset at any time.

The TxEN bit can be cleared at any time to disable the timer and stop the counting.

Figure 185. Continuous timer operation

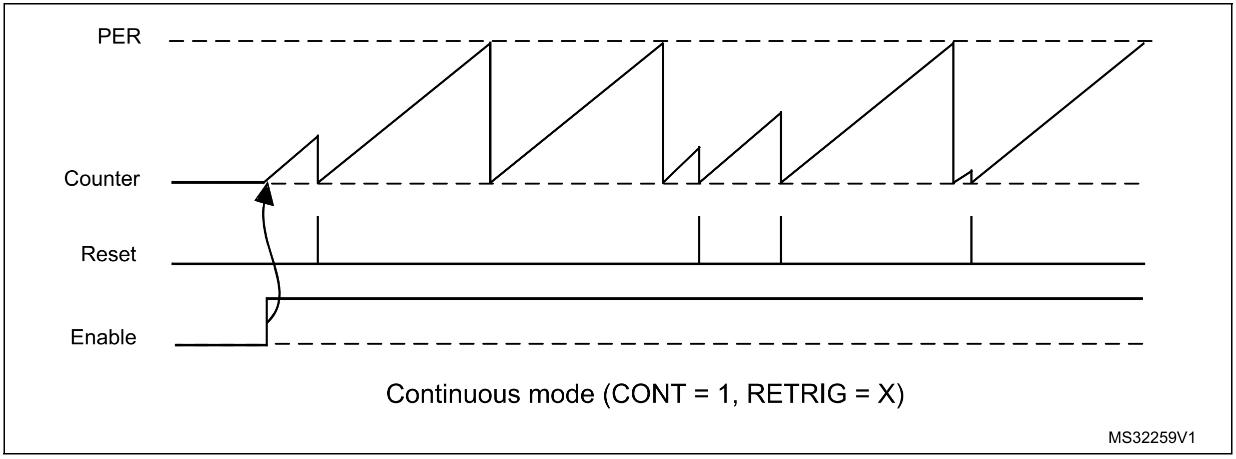
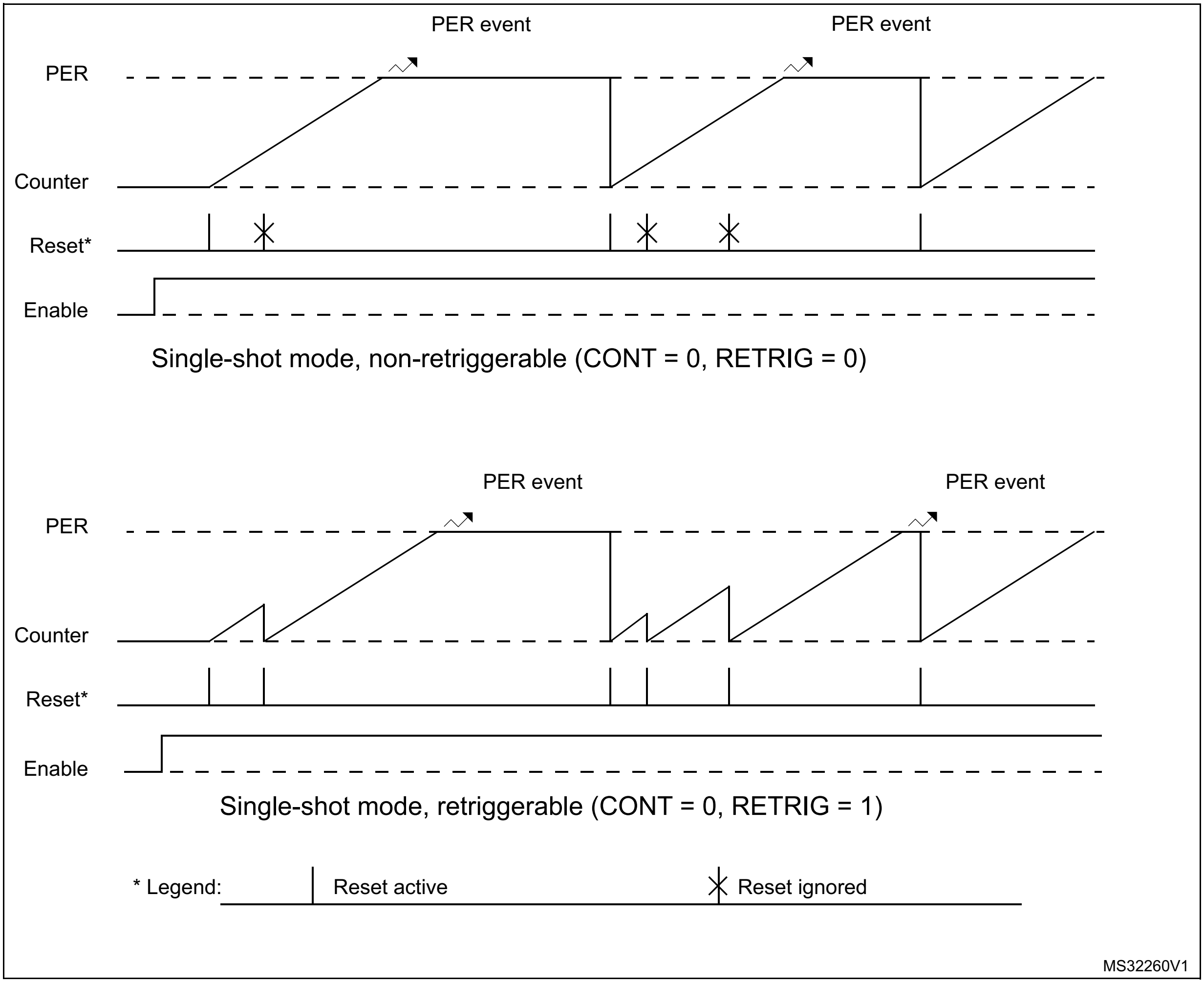


Figure 186. Single-shot timer operation



Roll-over event

A counter roll-over event is generated when the counter goes back to 0 after having reached the period value set in the HRTIM_PERxR register in continuous mode.

This event is used for multiple purposes in the HRTIM:

- To set/reset the outputs
- To trigger the register content update (transfer from preload to active)
- To trigger an IRQ or a DMA request
- To serve as a burst mode clock source or a burst start trigger
- As an ADC trigger
- To decrement the repetition counter

If the initial counter value is above the period value when the timer is started, or if a new period is set while the counter is already above this value, the counter is not reset: it overflows at the maximum period value and the repetition counter does not decrement.

Timer reset

The reset of the timing unit counter can be triggered by up to 30 events that can be selected simultaneously in the HRTIM_RSTxR register, among the following sources:

- The timing unit: compare 2, compare 4 and update (3 events)
- The master timer: reset and compare 1..4 (5 events)
- The external events EXTEVNT1..10 (10 events)
- All other timing units (e.g. timer B..F for timer A): compare 1, 2 and 4 (12 events)

Several events can be selected simultaneously to handle multiple reset sources. In this case, the multiple reset requests are ORed. When 2 counter reset events are generated within the same f_{HRTIM} clock cycle, the last counter reset is taken into account.

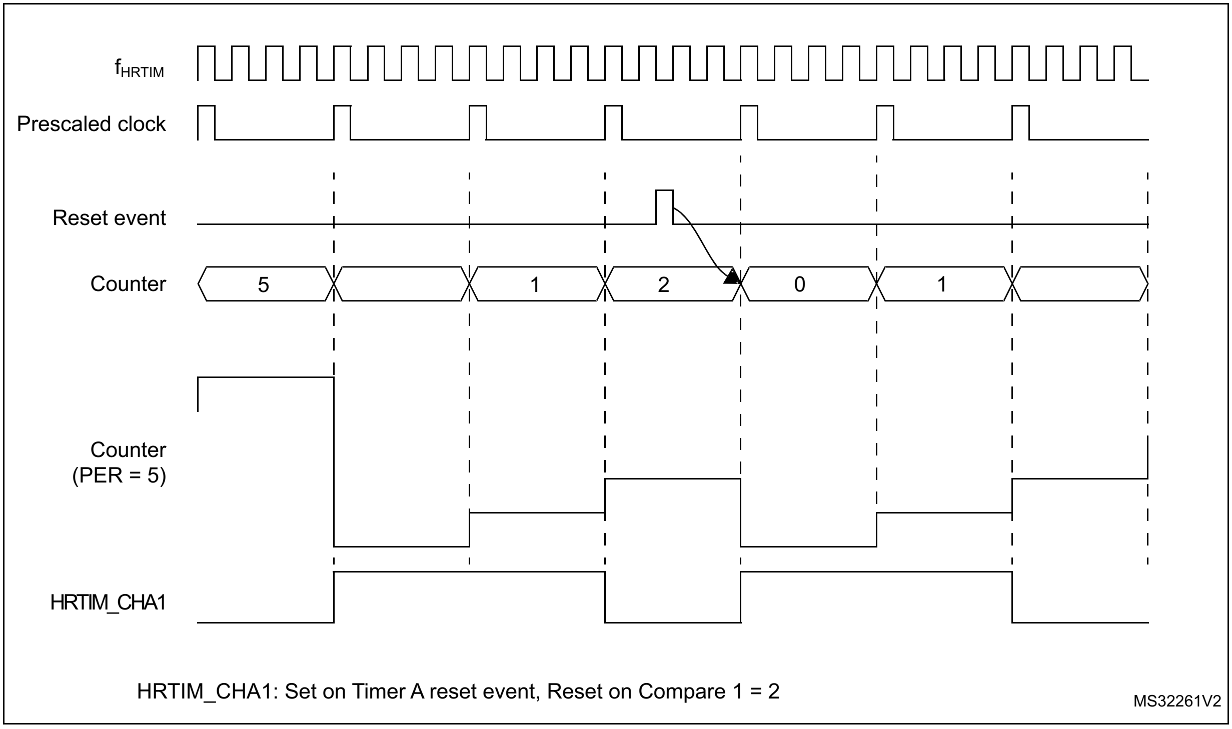
Additionally, it is possible to do a software reset of the counter using the TxRST bits in the HRTIM_CR2 register. These control bits are grouped into a single register to allow the simultaneous reset of several counters.

The reset requests are taken into account only once the related counters are enabled (TxCEN bit set).

When the f_{HRTIM} clock prescaling ratio is above 32 (counting period above f_{HRTIM}), the counter reset event is delayed to the next active edge of the prescaled clock. This allows to maintain a jitterless waveform generation when an output transition is synchronized to the reset event (typically a constant Ton time converter).

Figure 187 shows how the reset is handled for a clock prescaling ratio of 128 (f_{HRTIM} divided by 4).

Figure 187. Timer reset resynchronization (prescaling ratio above 32)



Repetition counter

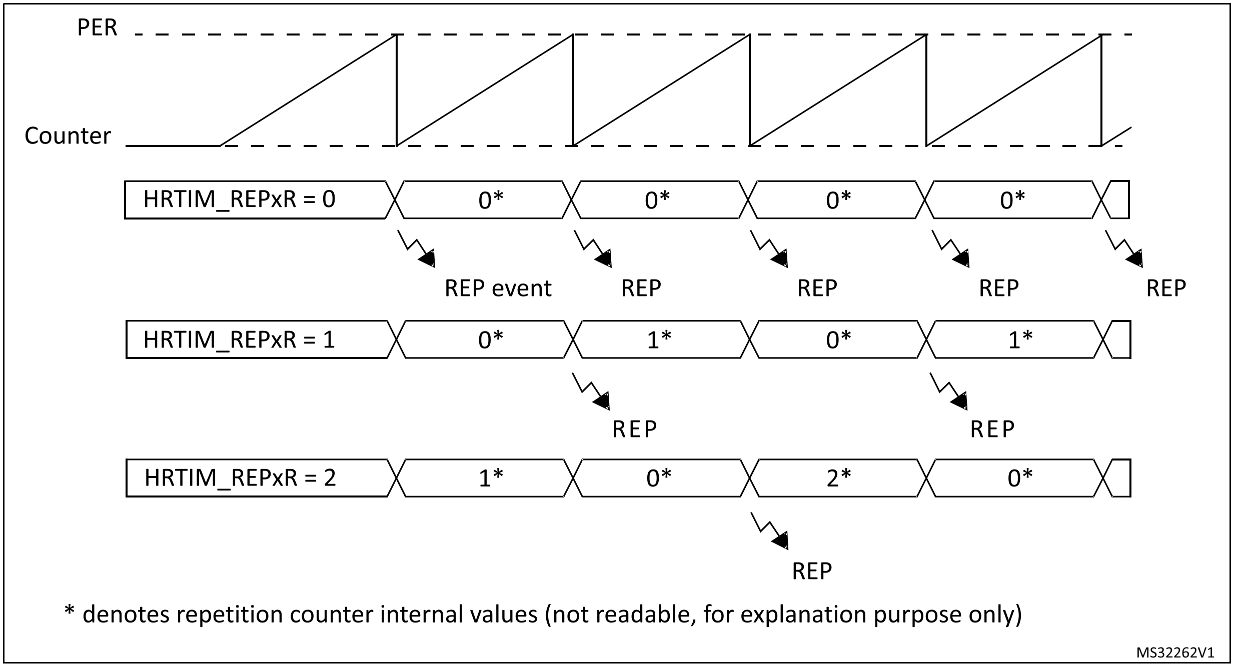
A common software practice is to have an interrupt generated when the period value is reached, so that the maximum amount of time is left for processing before the next period begins. The main purpose of the repetition counter is to adjust the period interrupt rate and off-load the CPU by decoupling the switching frequency and the interrupt frequency.

The timing units have a repetition counter. This counter cannot be read, but solely programmed with an auto-reload value in the HRTIM_REPxR register.

The repetition counter is initialized with the content of the HRTIM_REPxR register when the timer is enabled (TXCEN bit set). Once the timer has been enabled, any time the counter is cleared, either due to a reset event or due to a counter roll-over, the repetition counter is decreased. When it reaches zero, a REP interrupt or a DMA request is issued if enabled (REPIE and REPDE bits in the HRTIM_DIER register).

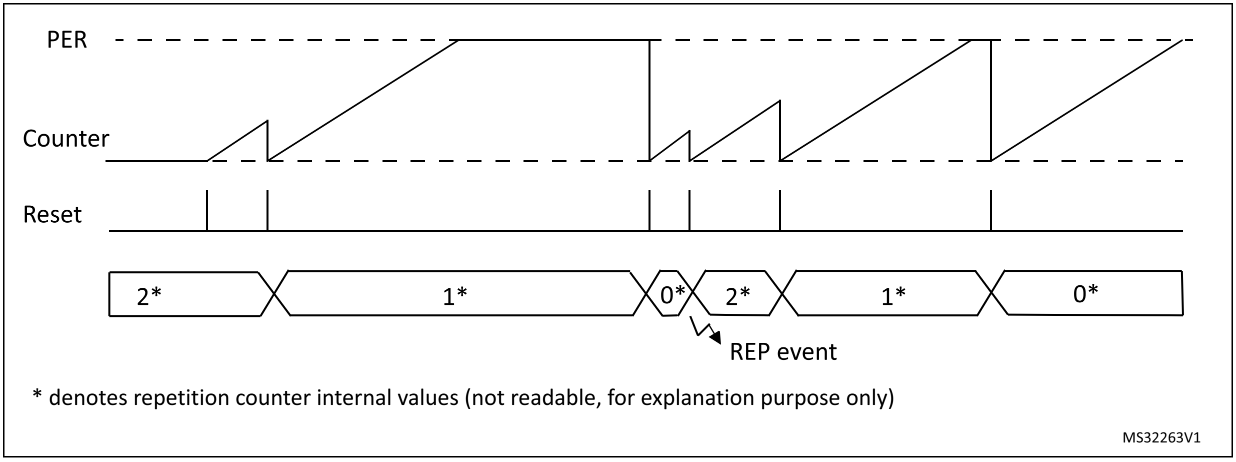
If the HRTIM_REPxR register is set to 0, an interrupt is generated for each and every period. For any value above 0, a REP interrupt is generated after (HRTIM_REPxR + 1) periods. *Figure 188* presents the repetition counter operation for various values, in continuous mode.

Figure 188. Repetition rate versus HRTIM_REPxR content in continuous mode



The repetition counter can also be used when the counter is reset before reaching the period value (variable frequency operation) either in continuous or in single-shot mode (*Figure 189* here-below). The reset causes the repetition counter to be decremented, at the exception of the very first start following counter enable (TxCEN bit set).

Figure 189. Repetition counter behavior in single-shot mode


PER
Counter
Reset
2*
1*
0*
2*
1*
0*
REP event
* denotes repetition counter internal values (not readable, for explanation purpose only)
MS32263V1

A reset or start event from the HRTIM_SCIN[3:1] source causes the repetition to be decremented as any other reset. However, in SYNCIN-started single-shot mode (SYNCSTRTx bit set in the HRTIM_TIMxCR register), the repetition counter is decremented only on the 1st reset event following the period. Any subsequent reset does not alter the repetition counter until the counter is re-started by a new request on HRTIM_SCIN[3:1] inputs.

Set / reset crossbar

A “set” event correspond to a transition to the output active state, while a “reset” event corresponds to a transition to the output inactive state.

The polarity of the waveform is defined in the output stage to accommodate positive or negative logic external components: an active level corresponds to a logic level 1 for a positive polarity (POLx = 0), and to a logic level 0 for a negative polarity (POLx = 1).

Each of the timing units handles the set/reset crossbar for two outputs. These 2 outputs can be set, reset or toggled by up to 32 events that can be selected among the following sources:

- The timing unit: period, compare 1..4, register update (6 events)
- The master timer: period, compare 1..4, HRTIM synchronization (6 events)
- All other timing units (e.g. timer B..F for timer A): TIMEVNT1..9 (9 events described in *Table 216*)
- The external events EXTEVNT1..10 (10 events)
- A software forcing (1 event)

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

The event sources are ORed and multiple events can be simultaneously selected.

Each output is controlled by two 32-bit registers, one coding for the set (HRTIM_SETxyR) and another one for the reset (HRTIM_RSTxyR), where x stands for the timing unit: A..F and y stands for the output 1or 2 (e.g. HRTIM_SETA1R, HRTIM_RSTC2R,...).

If the same event is selected for both set and reset, it toggles the output. It is not possible to toggle the output state more than one time per t_{HRTIM} period: in case of two consecutive toggling events within the same cycle, only the first one is considered.

The set and reset requests are taken into account only once the counter is enabled (TxCEN bit set), except if the software is forcing a request to allow the prepositioning of the outputs at timer start-up.

Table 216 summarizes the events from other timing units that can be used to set and reset the outputs. The number corresponds to the timer events (such as TIMEVNTx) listed in the register, and empty locations are indicating non-available events.

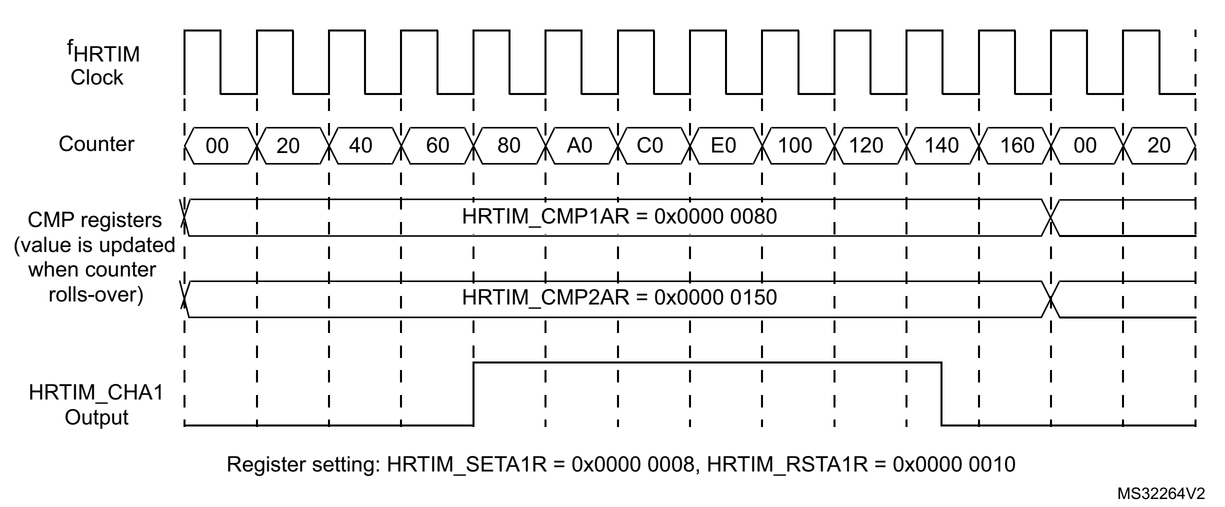
For instance, timer A outputs can be set or reset by the following events: timer B compare 1 and 2, timer C compare 2 and 3,... and timer E compare 3 is listed as TIMEVNT7 in HRTIM_SETA1R.

Table 216. Events mapping across timer A to F

Source		Timer A				Timer B				Timer C				Timer D				Timer E				Timer F			
		CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4
Destination	TA	-	-	-	-	1	2	-	-	-	3	4	-	5	6	-	-	-	-	7	8	-	-	-	9
	TB	1	2	-	-	-	-	-	-	-	-	3	4	-	-	5	6	7	8	-	-	-	-	9	-
	TC	-	1	2	-	-	3	4	-	-	-	-	-	-	5	-	6	-	-	7	8	-	9	-	-
	TD	1	-	-	2	-	3	-	4	-	-	-	5	-	-	-	-	6	-	-	7	8	-	9	-
	TE	-	-	-	1	-	-	2	3	4	5	-	-	6	7	-	-	-	-	-	-	-	-	8	9
	TF	-	-	1	-	2	-	-	3	4	-	-	5	-	-	6	7	-	8	9	-	-	-	-	-

Figure 190 represents how a PWM signal is generated using two compare events.

Figure 190. Compare events action on outputs: set on compare 1, reset on compare 2


fHRTIM Clock
Counter
00
20
40
60
80
A0
C0
E0
100
120
140
160
00
20
CMP registers (value is updated when counter rolls-over)
HRTIM_CMP1AR = 0x0000 0080
HRTIM_CMP2AR = 0x0000 0150
HRTIM_CHA1 Output
MS32264V2

Register setting: HRTIM_SETA1R = 0x0000 0008, HRTIM_RSTA1R = 0x0000 0010

Set / reset on update events

A set or reset event on update is done at low resolution. When CKPSC[2:0] < 5, the high-resolution delay is set to its maximum value so that a set/reset event on update always lags as compared to other compare set/reset events, with a jitter varying between 0 and 31/32 of a f_{HRTIM} clock period.

Half mode

This mode aims at generating square signal with fixed 50% duty cycle and variable frequency (typically for converters using resonant topologies). It allows to have the duty cycle automatically forced to half of the period value when a new period is programmed.

This mode is enabled by writing HALF bit to 1 in the HRTIM_TIMxCR register. When the HRTIM_PERxR register is written, it causes an automatic update of the compare 1 value with HRTIM_PERxR/2 value.

The output on which a square wave is generated must be programmed to have one transition on CMP1 event, and one transition on the period event, as follows:

– HRTIM_SETxyR = 0x0000 0008, HRTIM_RSTxyR = 0x0000 0004, or

– HRTIM_SETxyR = 0x0000 0004, HRTIM_RSTxyR = 0x0000 0008

The HALF mode overrides the content of the HRTIM_CMP1xR register. The access to the HRTIM_PERxR register only causes compare 1 internal register to be updated. The user-accessible HRTIM_CMP1xR register is not updated with the HRTIM_PERxR / 2 value.

When the preload is enabled (PREEN = 1, MUDIS, TxUDIS), compare 1 active register is refreshed on the update event. If the preload is disabled (PREEN= 0), compare 1 active register is updated as soon as HRTIM_PERxR is written.

The period must be greater than or equal to 6 periods of the f_{HRTIM} clock (0xC0 if CKPSC[2:0] = 0, 0x60 if CKPSC[2:0] = 1, 0x30 if CKPSC[2:0] = 2,...) when the HALF mode is enabled.

Interleaved mode

This mode complements the Half mode and helps the implementation of interleaved topologies.

It allows to re-compute automatically the content of compare registers when the HRTIM_PERxR value is updated.

The selection is done using the HALF bit and the IL[1:0] bits in HRTIM_MCR and HRTIM_TIMxCR registers, as shown on the *Table 217* below.

Table 217. Interleaved mode selection

HALF bit	INTLVD [1:0] bits	Mode
0	00	Disabled
0	01	Triple interleaved (120°)
0	10	Quad interleaved (90°)
0	11	Reserved
1	xx	Dual interleaved (180°)

Table 218 gives the compare values for the 3 available modes. The content of the compare registers is overridden. The corresponding compare events can be used to trigger an output set / reset or to reset a slave timer.

Table 218. Compare 1..3 values in interleaved mode

Mode	Dual interleaved 180°	Triple interleaved 120°	Quad interleaved 90°
CMP1 value	PERxR/2	PERxR/3	PERxR/4
CMP2 value	Not affected	2x (PERxR/3)	PERxR/2
CMP3 value	Not affected	Not affected	3x (PERxR/4)

Note: In half and interleaved modes, the compare registers are controlled by hardware and writing them has no effect. However the written value is stored in the preload register and becomes active on the update event following the exit of these modes.

Note: The triple and quad interleaved modes must not be used simultaneously with other modes using CMP2 (dual channel dac trigger and triggered-half modes).

Null duty cycle exception case

The high-resolution behavior is not supported for pulses narrower than 3 t_{HRTIM} periods (see *Section 27.3.7: Set / reset events priorities and narrow pulses management*) and any value strictly below 3 periods of the f_{HRTIM} clock (that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...) in the HRTIM_TIMxCMPy register is forbidden (see *27.5.19: HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F)*).

However, it is possible to skip an output pulse and have a null duty cycle by simply writing a null value in the following two registers: HRTIM_TIMxCMP1 and HRTIM_TIMxCMP3, if and only if the following conditions are met:

- the output SET event is generated by the PERIOD event
- the output RESET if generated by the compare 1 (respectively compare 3) event
- the compare 1 (compare 3) event is active within the timer unit itself, and not used for other timing units

For any other use case, this can be done by programming the SET and RESET events with the very same compare values, above 3 periods of the f_{HRTIM} clock. In this case, the output is reset forced (following the set/reset priority scheme defined in the *Section 27.3.7: Set / reset events priorities and narrow pulses management*).

Swap mode

This mode allows to swap the two outputs with a single bit access: the output 1 signal is connected to the output 2 pin and the output 2 signal is connected to output 1 pin. The output swap is triggered with the SWPx bits in the HRTIM_CR2 register and is effective on the next update event.

The outputs are swapped prior to the set/reset crossbar unit, as following:

- if SWPx = 0, HRTIM_SETx1R and HRTIM_RSTx1R are coding for the output 1, HRTIM_SETx2R and HRTIM_RSTx2R are coding for the output 2
- if SWPx = 1, HRTIM_SETx1R and HRTIM_RSTx1R are coding for the output 2, HRTIM_SETx2R and HRTIM_RSTx2R are coding for the output 1

The swap mode is only affecting the preload register, and not the active registers.

Note: The preload mode must be enabled when using the swap mode.

Consequently, it does not modify the auxiliary outputs in parallel with the regular outputs going to the output stage (see Section 27.3.18 for details). They provide the following internal status, events and signals:

- O1CPY, O2CPY, SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests
- Capture triggers upon output set/reset (TA2, TB2, TC2, TD2, TE2, TF2)
- External event filters generated with a Tx2 output copy

For instance the SETx1 flag is related to the output 1 when SWP = 0 and is related to the output 2 when SWPx = 1.

Similarly, the swap mode does not change the attribution of control bits in the HRTIM_OUTxR register (DIDLx, CHPx, FAULTx[1:0], IDLESx, POLx bits). For instance, the POL1 bit controls the output 1 polarity whatever the SWP bit value.

Note: The SWPx bits are ignored in push-pull mode (PSHPLL = 1 in the HRTIM_TIMxCR register).

Capture

The timing unit has the capability to capture the counter value, triggered by internal and external events. The purpose is to:

- measure events arrival timings or occurrence intervals
- update compare 2 and compare 4 values in auto-delayed mode (see *Section : Auto-delayed mode*).

The capture is done with f_{HRTIM} resolution: for a clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the register are not significant (read as 0).

The timer has 2 capture registers: HRTIM_CPT1xR and HRTIM_CPT2xR. The capture triggers are programmed in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers.

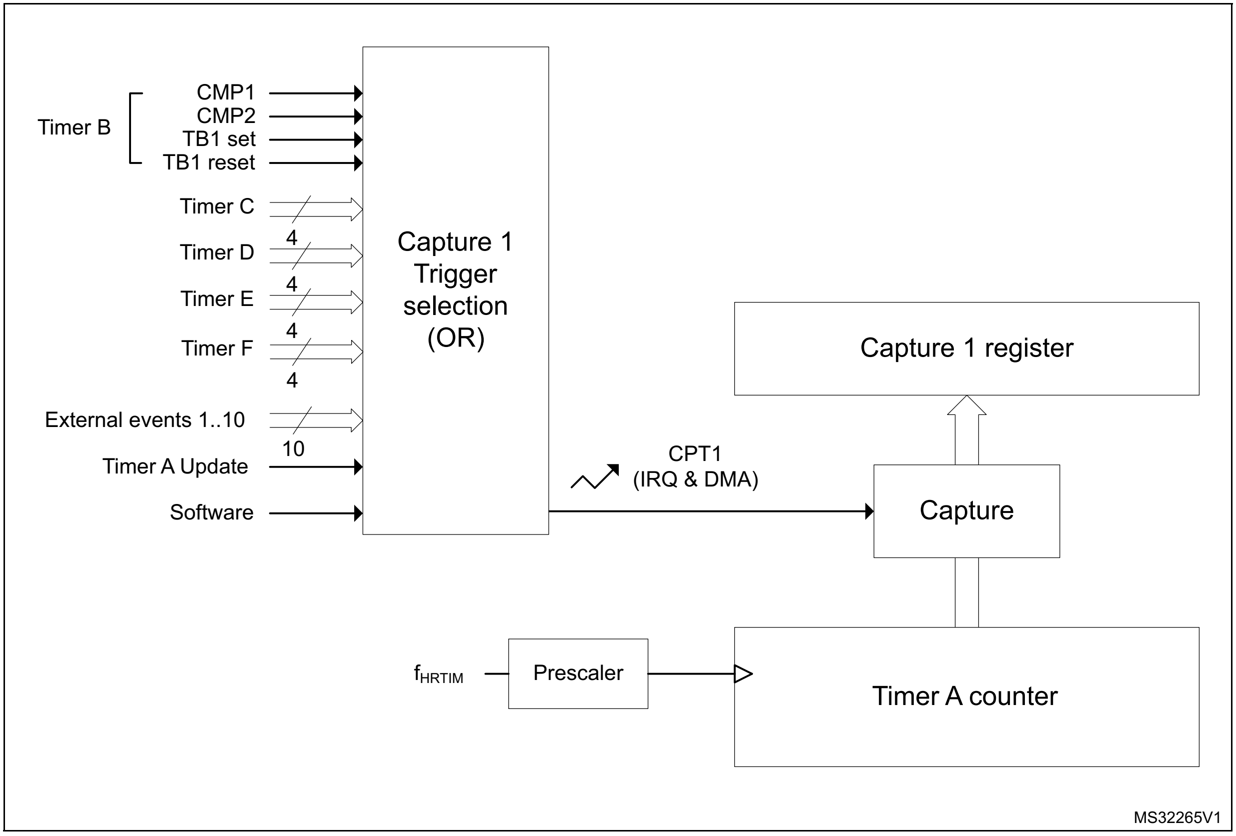
The capture of the timing unit counter can be triggered by up to 28 events that can be selected simultaneously in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers, among the following sources:

- The external events, EXTEVNT1..10 (10 events)
- All other timing units (e.g. timer B..F for timer A): compare 1, 2 and output 1 set/reset events (16 events)
- The timing unit: update (1 event)
- A software capture (1 event)

Several events can be selected simultaneously to handle multiple capture triggers. In this case, the concurrent trigger requests are ORed. The capture can generate an interrupt or a DMA request when CPTxIE and CPTxDE bits are set in the HRTIM_TIMxDIER register.

Over-capture is not prevented by the circuitry: a new capture is triggered even if the previous value was not read, or if the capture flag was not cleared.

Figure 191. Timer A timing unit capture circuitry



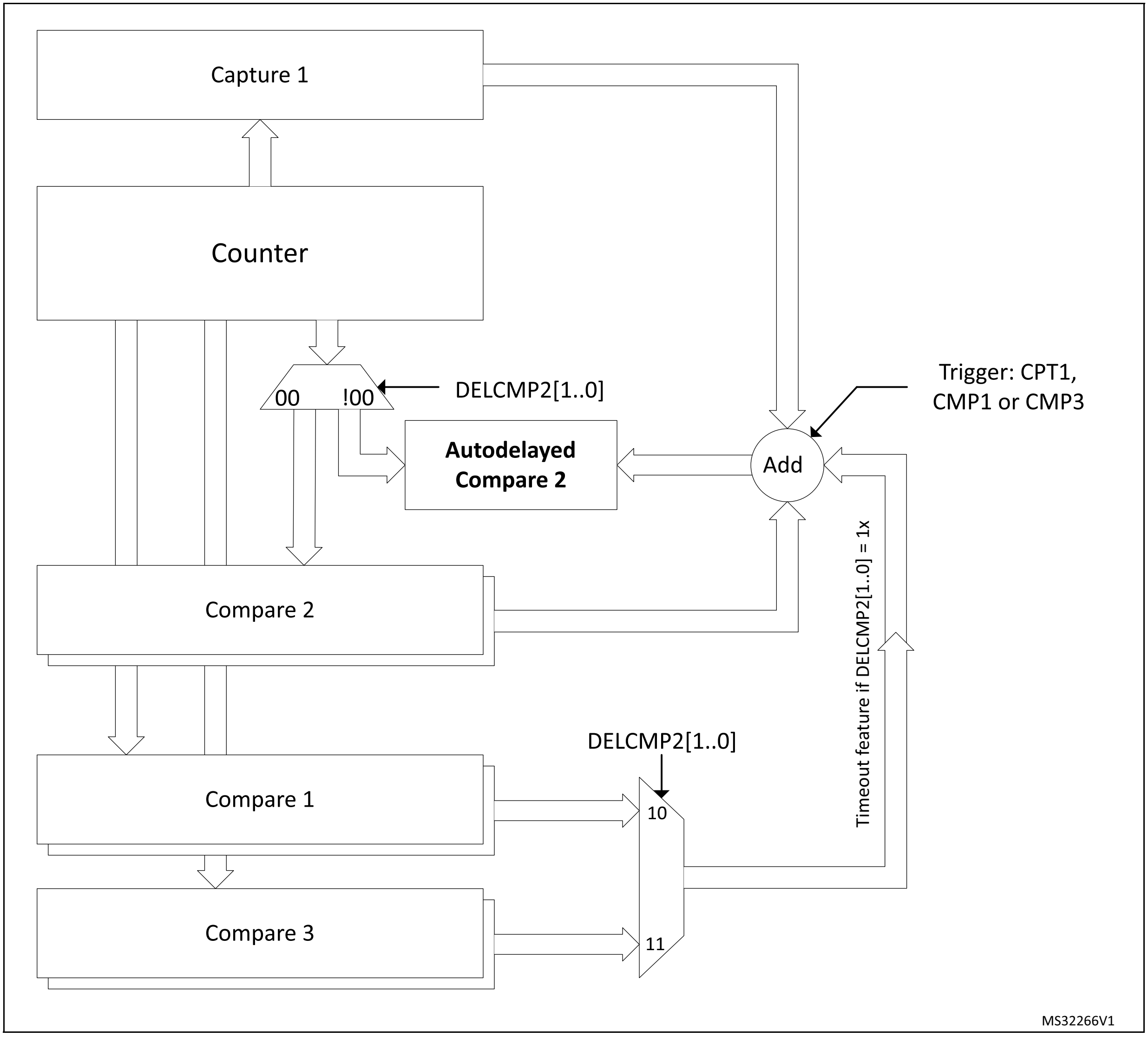
Auto-delayed mode

This mode allows to have compare events generated relatively to capture events, so that for instance an output change can happen with a programmed timing following a capture. In this case, the compare match occurs independently from the timer counter value. It enables the generation of waveforms with timings synchronized to external events without the need of software computation and interrupt servicing.

As long as no capture is triggered, the content of the HRTIM_CMPxR register is ignored (no compare event is generated when the counter value matches the compare value. Once the capture is triggered, the compare value programmed in HRTIM_CMPxR is summed with the captured counter value in HRTIM_CPTxyR, and it updates the internal auto-delayed compare register, as seen on *Figure 192*. The auto-delayed compare register is internal to the timing unit and cannot be read. The HRTIM_CMPxR preload register is not modified after the calculation.

This feature is available only for compare 2 and compare 4 registers. compare 2 is associated with capture 1, while compare 4 is associated with capture 2. HRTIM_CMP2xR and HRTIM_CMP4xR compares cannot be programmed with a value below 3 f_{HRTIM} clock periods, as in the regular mode.

Figure 192. Auto-delayed overview (compare 2 only)



The auto-delayed compare is only valid from the capture up to the period event: once the counter has reached the period value, the system is re-armed with compare disabled until a capture occurs.

DELCMP2[1:0] and DELCMP4[1:0] bits in HRTIM_TIMxCR register allow to configure the auto-delayed mode as follows:

- 00
  Regular compare mode: HRTIM_CMP2xR and HRTIM_CMP4xR register contents are directly compared with the counter value.
- 01
  Auto-delayed mode: compare 2 and compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event.
- 10 or 11
  Auto-delayed mode with timeout: compare 2 and compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event or after a compare 1

match (DELCMPx[1:0]= 10) or a compare 3 match (DELCMPx[1:0]= 11) to have a timeout function if capture 1/2 event is missing.

When the capture occurs, the comparison is done with the (HRTIM_CMP2/4xR + HRTIM_CPT1/2xR) value. If no capture is triggered within the period, the behavior depends on the DELCMPx[1:0] value:

- DELCMPx[1:0] = 01: the compare event is not generated
- DELCMPx[1:0] = 10 or 11: the comparison is done with the sum of the 2 compares (for instance HRTIM_CMP2xR + HRTIM_CMP1xR). The captures are not taken into account if they are triggered after CMPx + CMP1 (resp. CMPx + CMP3).

The captures are enabled again at the beginning of the next PWM period.

If the result of the auto-delayed summation is above 0xFFFF (overflow), the value is ignored and no compare event is generated until a new period is started.

Note: DELCMPx[1:0] bitfield must be reset when reprogrammed from one value to the other to re-initialize properly the auto-delayed mechanism, for instance:

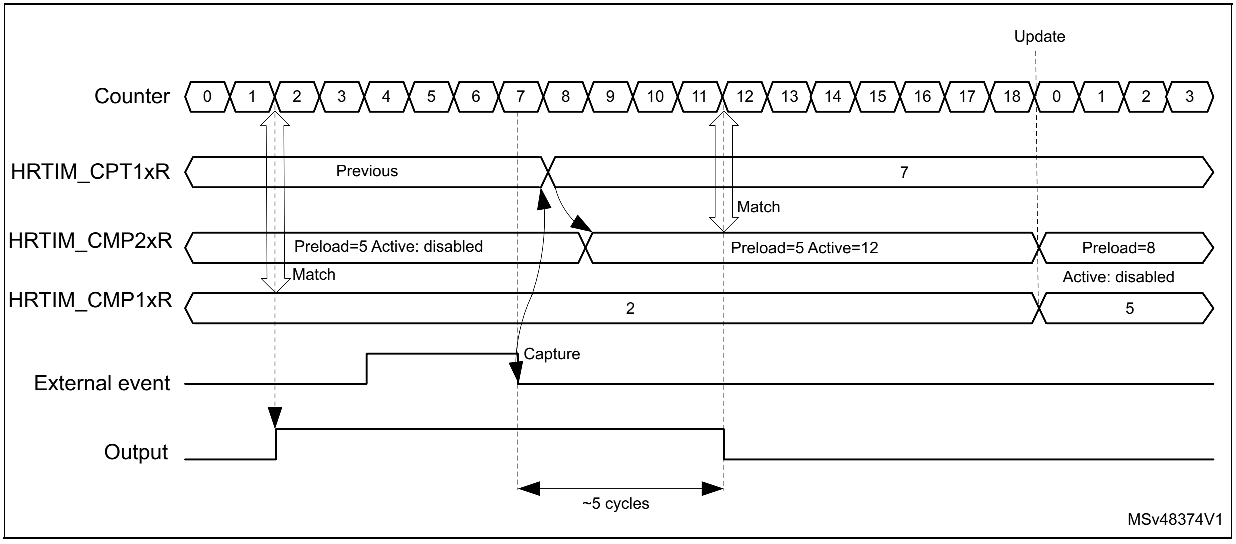
- DELCMPx[1:0] = 10
- DELCMPx[1:0] = 00
- DELCMPx[1:0] = 11

As an example, *Figure 193* shows how the following signal is generated:

- Output set when the counter is equal to compare 1 value
- Output reset 5 cycles after a falling edge on a given external event

Note: To simplify the figure, the high-resolution is not used in this example (CKPSC[2:0] = 101), thus the counter is incremented at the f_{HRTIM} rate. Similarly, the external event signal is shown without any resynchronization delay: practically, there is a delay of 1 to 2 f_{HRTIM} clock periods between the falling edge and the capture event due to an internal resynchronization stage which is necessary to process external input signals.

Figure 193. Auto-delayed compare



A regular compare channel (e.g. compare 1) is used for the output set: as soon as the counter matches the content of the compare register, the output goes to its active state.

A delayed compare is used for the output reset: the compare event can be generated only if a capture event has occurred. No event is generated when the counter matches the delayed compare value (counter = 4). Once the capture event has been triggered by the external event, the content of the capture register is summed to the delayed compare value to have the new compare value. In the example, the auto-delayed value 4 is summed to the capture equal to 7 to give a value of 12 in the auto-delayed compare register. From this time on, the compare event can be generated and happens when the counter is equal to 12, causing the output to be reset.

Overcapture management in auto-delayed mode

Overcapture is prevented when the auto-delayed mode is enabled (DELCMPx[1:0] = 01, 10, 11).

When multiple capture requests occur within the same counting period, only the first capture is taken into account to compute the auto-delayed compare value. A new capture is possible only:

- Once the auto-delayed compare has matched the counter value (compare event)
- Once the counter has rolled over (period)
- Once the timer has been reset

Changing auto-delayed compare values

When the auto-delayed compare value is preloaded (PREEN bit set), the new compare value is taken into account on the next coming update event (for instance on the period event), regardless of when the compare register was written and if the capture occurred (see *Figure 193*, where the delay is changed when the counter rolls over).

When the preload is disabled (PREEN bit reset), the new compare value is taken into account immediately, even if it is modified after the capture event has occurred, as per the example below:

1. At t1, DELCMP2 = 1.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, a capture event occurs capturing the value CPTR1 = 0x20. => comparison enabled, compare value = 0x60
4. At t4, CMP2_act = 0x100 (before the counter reached value CPTR1 + 0x40) => comparison still enabled, new compare value = 0x120
5. At t5, the counter reaches the period value => comparison disabled, cmp2_act = 0x100

Similarly, if the CMP1(CMP3) value changes while DELCMPx = 10 or 11, and preload is disabled:

1. At t1, DELCMP2 = 2.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, CMP3 event occurs - CMP3_act = 0x50 before capture 1 event occurs => comparison enabled, compare value = 0x90
4. At t4, CMP3_act = 0x100 (before the counter reached value 0x90) => comparison still enabled, compare 2 event occurs at = 0x140

Triggered-half mode

The purpose of this mode is to allow the synchronization of 2 interleaved converters that have variable frequency operation and require a 180° phase-shift. The basic principle is to have a master-slave system.The slave converter synchronization is continuously adjusted based on the previous switching period of the master converter.

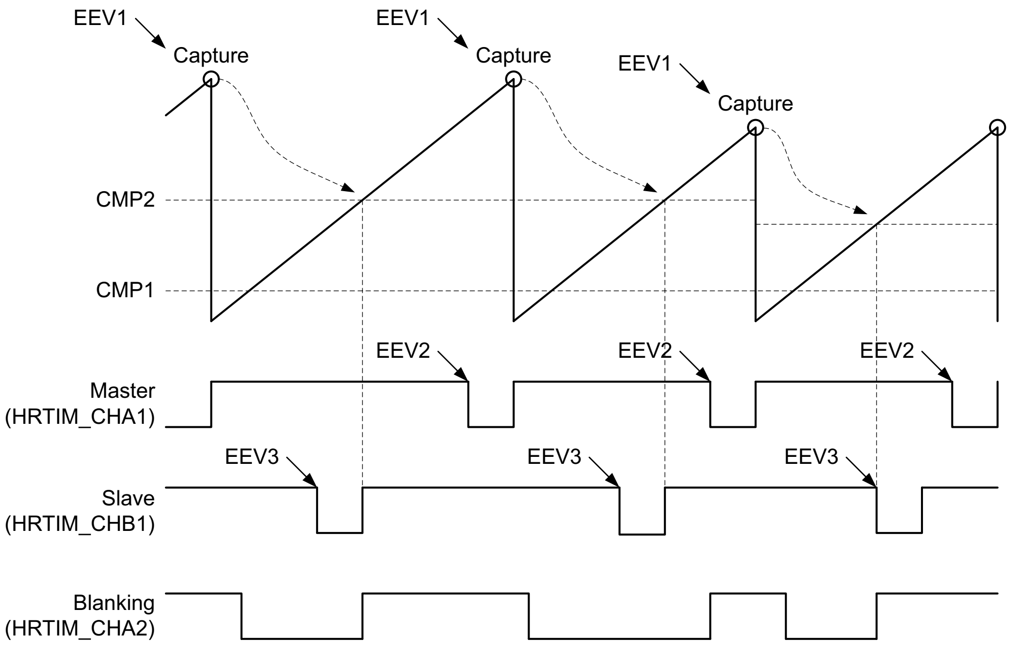
This is done using the capture unit. The switching period of the master converter is captured, divided by 2 and then stored in the compare 2 register by hardware. The compare 2 register contains a value equal to half of the captured period, which is the master converter switching period. The compare 2 event can then be used to trigger a second timing unit that manages the slave converter.

This mode is enabled by setting the TRGHLF bit in the HRTIM_TIMxCR2 register. This bit cannot be changed once the timer is operating (TxEN bit set).

The triggered-half mode must not be used simultaneously with other modes using CMP2 (dual channel dac trigger, interleaved and balanced idle modes).

The initial value CMP2 can be written by the user, but is ignored once the first capture is triggered. The preload mechanism is disabled for CMP2 when the TRGHLF bit is reset.

Figure 194. Triggered-half mode example



HRTIM_CHA1
Set on EEV1 (EEV1 triggers Capture1)
Reset on EEV2
EEV1 Blanking on TIMA_CMP1 (avoid frequency run-away)
HRTIM_CHB1
Set on CMP2 (triggered-half) or EEV1 blanking by TIMA_TA2
(TB1 is set if EEV1 occurs between CMP1 and CMP2, else on CMP2)
Reset on EEV3
HRTIM_CHA2
Set on TA_CMP1
Reset on TA_CMP2

Note: In triggered half mode, the compare2 register is controlled by hardware and writing it has no effect. However the written value is stored in the preload register and becomes active on the update event following the exit of this mode.

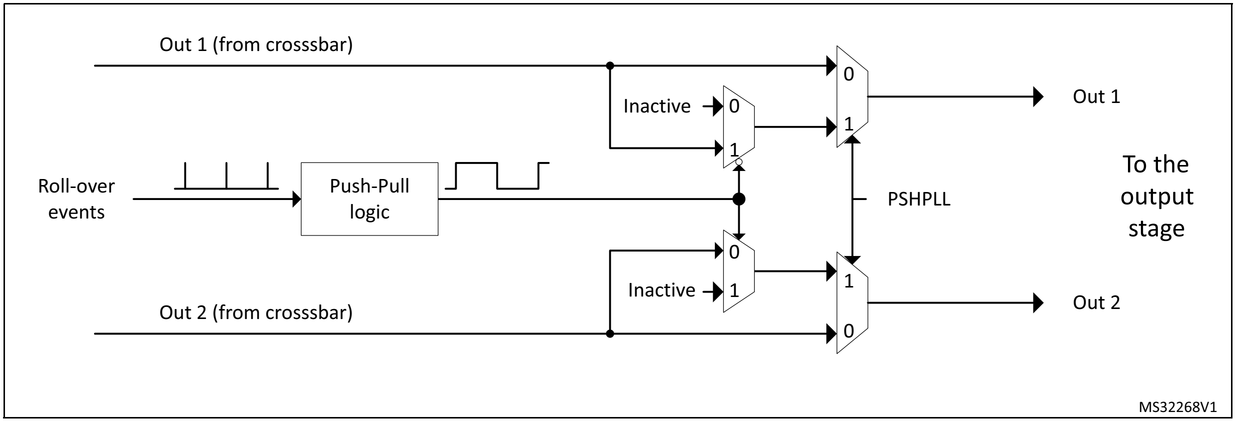
Push-pull mode

This mode primarily aims at driving converters using push-pull topologies. It also needs to be enabled when the delayed idle protection is required, typically for resonant converters (refer to *Section 27.3.10: Delayed protection*).

The push-pull mode is enabled by setting PSHPLL bit in the HRTIM_TIMxCR register.

It applies the signals generated by the crossbar to output 1 and output 2 alternatively, on the period basis, maintaining the other output to its inactive state. The redirection rate (push-pull frequency) is defined by the timer's period event, as shown on *Figure 195*. The push-pull period is twice the timer counting period.

Figure 195. Push-pull mode block diagram



The push-pull mode is available when the timer operates in continuous mode and in single-shot mode. It is necessary to disable the timer to stop a push-pull operation and to reset the counter before re-enabling it.

To get a correct behavior, the event selected as source of the counter reset must be also selected for setting (or resetting) the output. It must set the output if the output is set on period, else it must reset it. If it is not done, the output switching from its inactive period to its active period may be incorrect (may unexpectedly rise or may unexpectedly stay low).

The signal shape is defined using HRTIM_SETxyR and HRTIM_RSTxyR for both outputs. It is necessary to have HRTIM_SETx1R = HRTIM_SETx2R and HRTIM_RSTx1R = HRTIM_RSTx2R to have both outputs with identical waveforms and to achieve a balanced operation. Still, it is possible to have different programming on both outputs for other uses.

The CPPSAT status bit in HRTIM_TIMxISR indicates on which output the signal is currently active. CPPSTAT is reset when the push-pull mode is disabled.

In the example given on *Figure 196*, the timer internal waveform is defined as follows:

- Output set on period event
- Output reset on compare 1 match event

Figure 196. Push-pull mode example

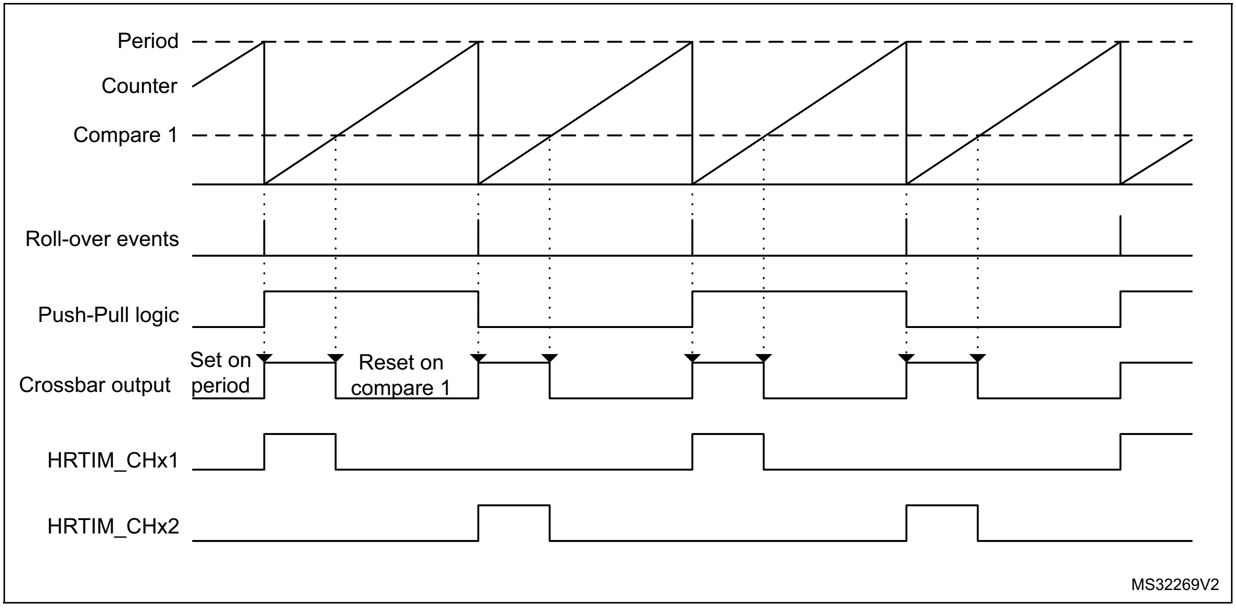
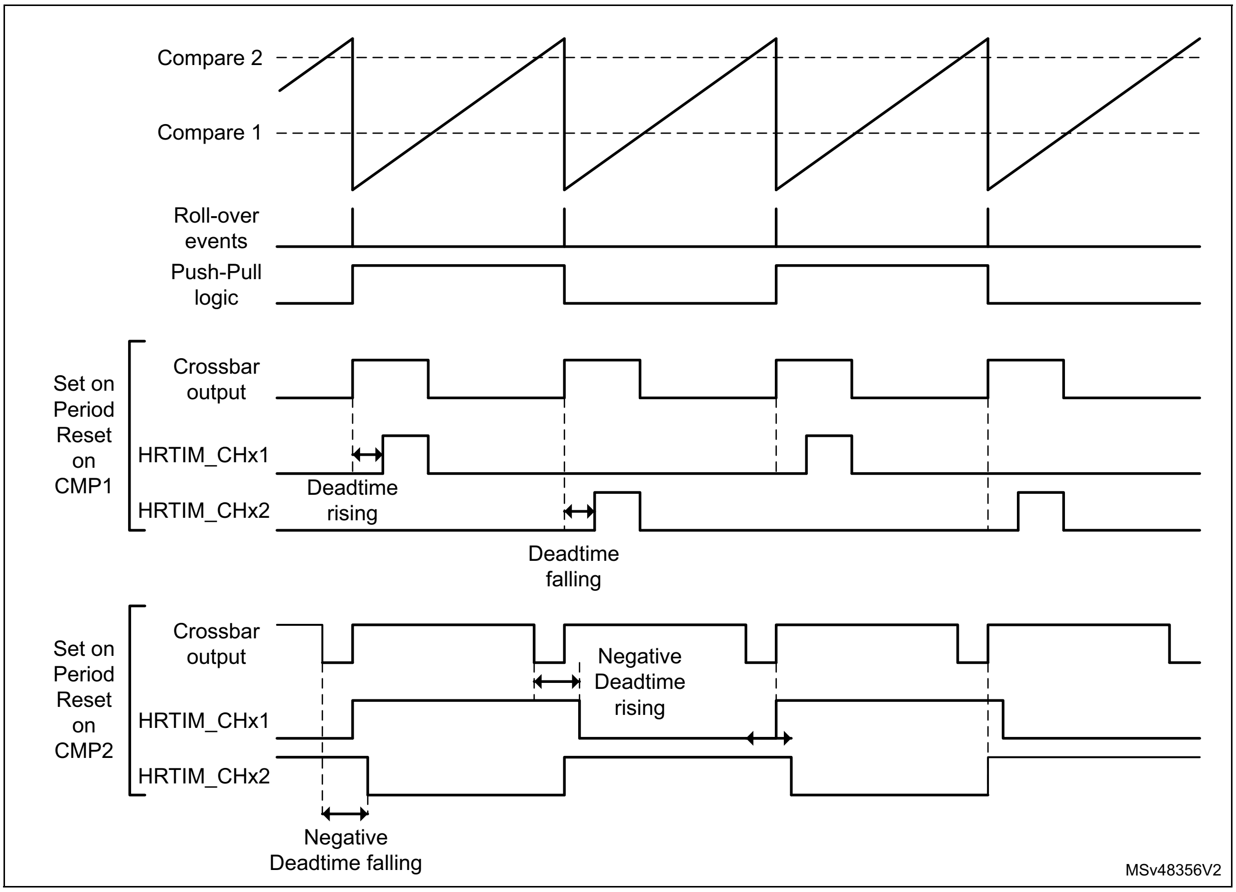


Figure 197 shows how the deadtime is inserted in push-pull mode, for both positive and negative deadtimes. In this case, the outputs are not complemented any more, and the deadtime is applied on each output individually (both ouput 1 and output 2 of the crossbar are used).

Figure 197. Push-Pull with deadtime



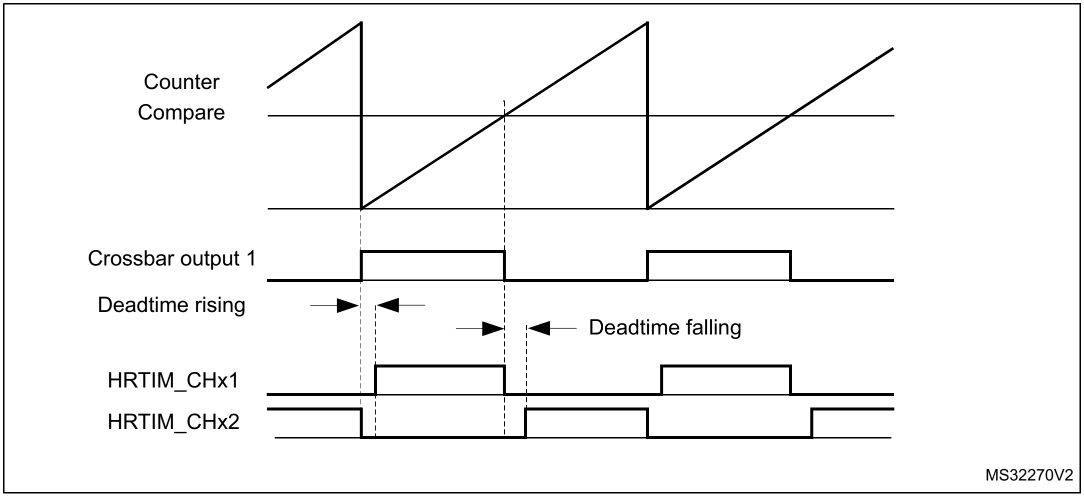
Deadtime

A deadtime insertion unit allows to generate a couple of complementary signals from a single reference waveform, with programmable delays between active state transitions. This is commonly used for topologies using half-bridges or full bridges. It simplifies the software: only 1 waveform is programmed and controlled to drive two outputs.

The Dead time insertion is enabled by setting DTEN bit in HRTIM_OUTxR register. The complementary signals are built based on the reference waveform defined for output 1, using HRTIM_SETx1R and HRTIM_RSTx1R registers: HRTIM_SETx2R and HRTIM_RSTx2R registers are not significant when DTEN bit is set.

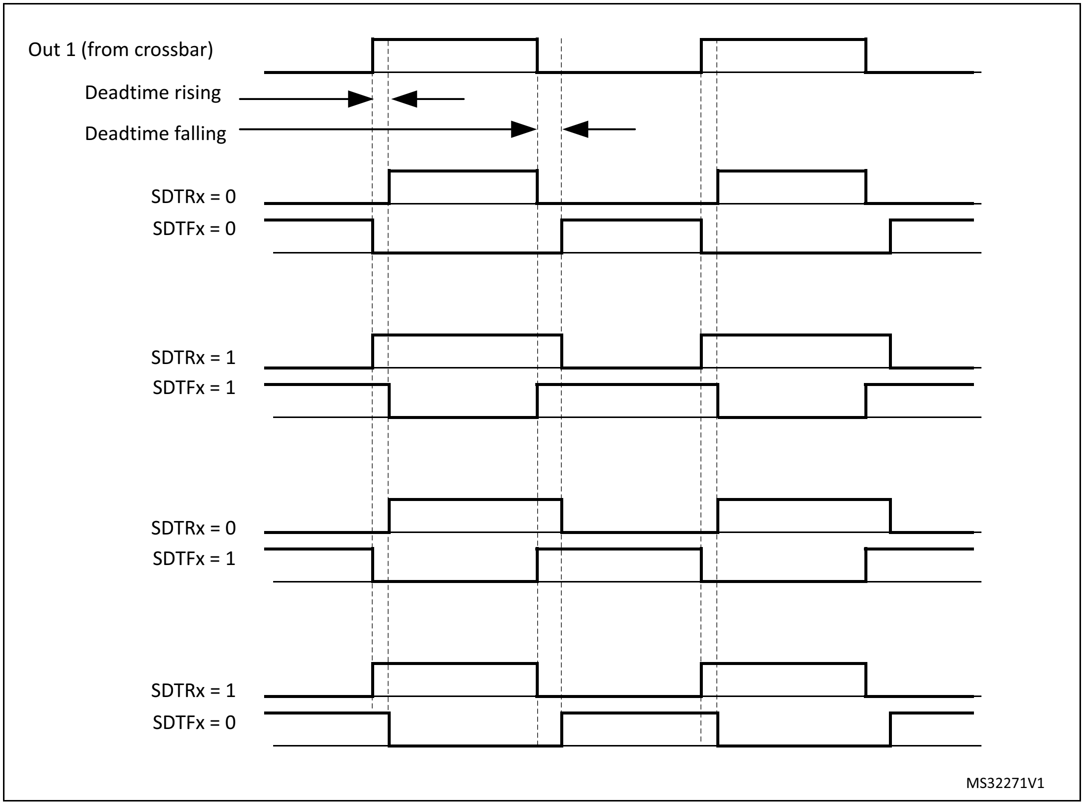
Two deadtimes can be defined in relationship with the rising edge and the falling edge of the reference waveform, as in *Figure 198*.

Figure 198. Complementary outputs with deadtime insertion



Negative deadtime values can be defined when some control overlap is required. This is done using the deadtime sign bits (SDTFx and SDTRx bits in HRTIM_DTxR register). *Figure 199* shows complementary signal waveforms depending on respective signs.

Figure 199. Deadtime insertion versus deadtime sign (1 indicates negative deadtime)



The deadtime values are defined with DTFx[8:0] and DTRx[8:0] bitfields and based on a specific clock prescaled according to DTPRSC[2:0] bits, as follows:

t_{DTx} = +/- DTx[8:0] x t_{DTG}

where x is either R or F and $t_{DTG} = (2^{(DTPRSC[2:0])}) \times (t_{HRTIM} / 8)$.

Table 219 gives the resolution and maximum absolute values depending on the prescaler value.

Table 219. Deadtime resolution and max absolute values

DTPRSC[2:0]	t_{DTG}	t_{DTx} max	f_{HRTIM}= 170 MHz	
			t_{DTG} (ns)	$\|t_{DTx}\|$ max (µs)
000	t_{HRTIM} / 8	511 * t_{DTG}	0.73	0.38
001	t_{HRTIM} / 4		1.47	0.75
010	t_{HRTIM} / 2		2.94	1.50
011	t_{HRTIM}		5.89	3.01
100	2 * t_{HRTIM}		11.76	6.01
101	4 * t_{HRTIM}		23.53	12.03
110	8 * t_{HRTIM}		47.06	24.04
111	16 * t_{HRTIM}		94.12	48.10

Figure 200 to *Figure 203* present how the deadtime generator behaves for reference waveforms with pulsewidth below the deadtime values, for all deadtime configurations.

Figure 200. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 0)

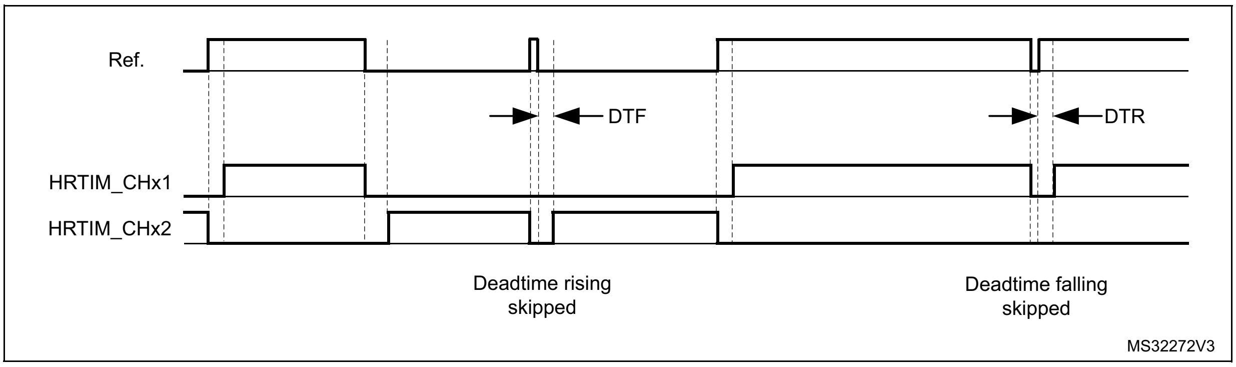


Figure 201. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 1)

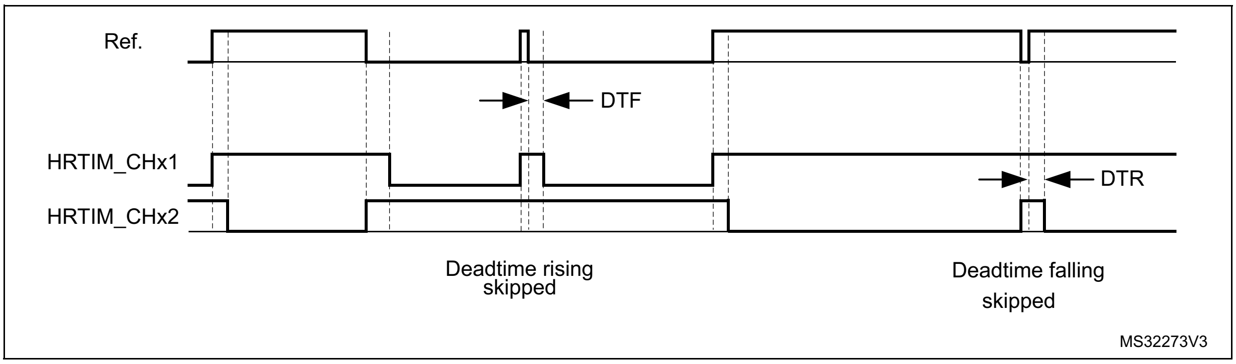


Figure 202. Complementary outputs for low pulsewidth (SDTRx = 0, SDTFx = 1)

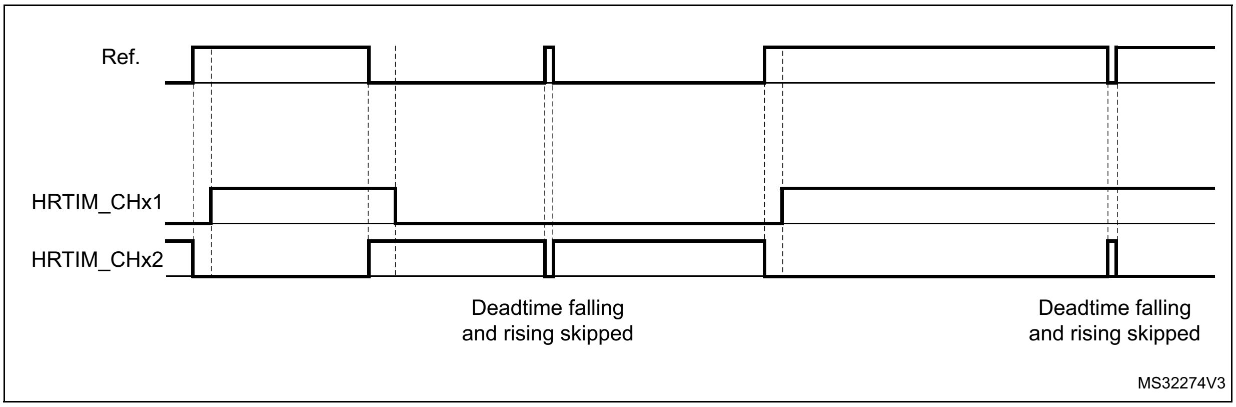
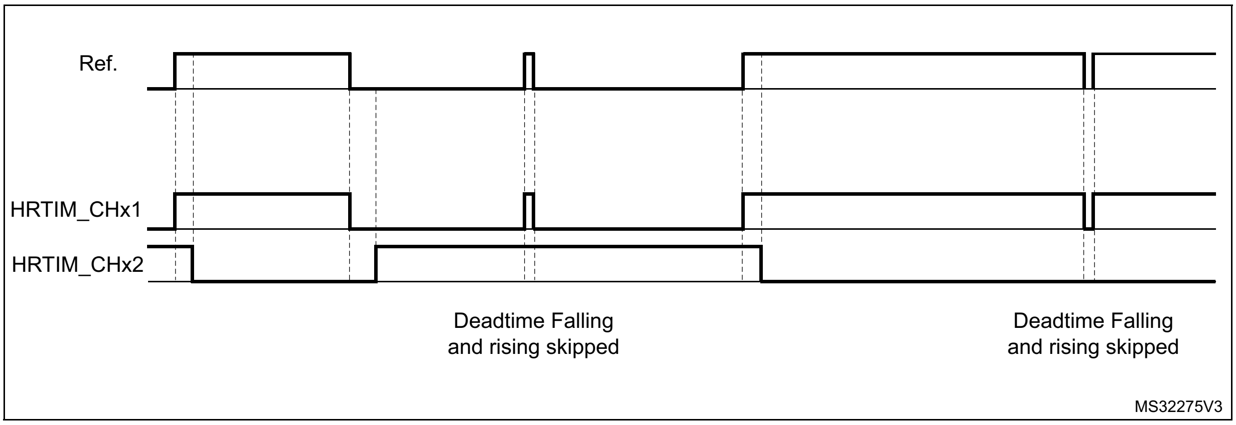

Ref.
HRTIM_CHx1
HRTIM_CHx2
Deadtime falling and rising skipped
Deadtime falling and rising skipped
MS32274V3

Figure 203. Complementary outputs for low pulsewidth (SDTRx = 1, SDTFx=0)


Ref.
HRTIM_CHx1
HRTIM_CHx2
Deadtime Falling and rising skipped
Deadtime Falling and rising skipped
MS32275V3

For safety purposes, it is possible to prevent any spurious write into the deadtime registers by locking the sign and/or the value of the deadtime using DTFLKx, DTRLKx, DTFSLKx and DTRSLKx. Once these bits are set, the related bits and bitfields are becoming read only until the next system reset.

Caution: DTEN bit must not be changed in the following cases:

- When the timer is enabled (TxEN bit set)
- When the timer outputs are set/reset by another timer (while TxEN is reset)

Otherwise, an unpredictable behavior would result.

It is therefore necessary to disable the timer (TxCEN bit reset) and have the corresponding outputs disabled.

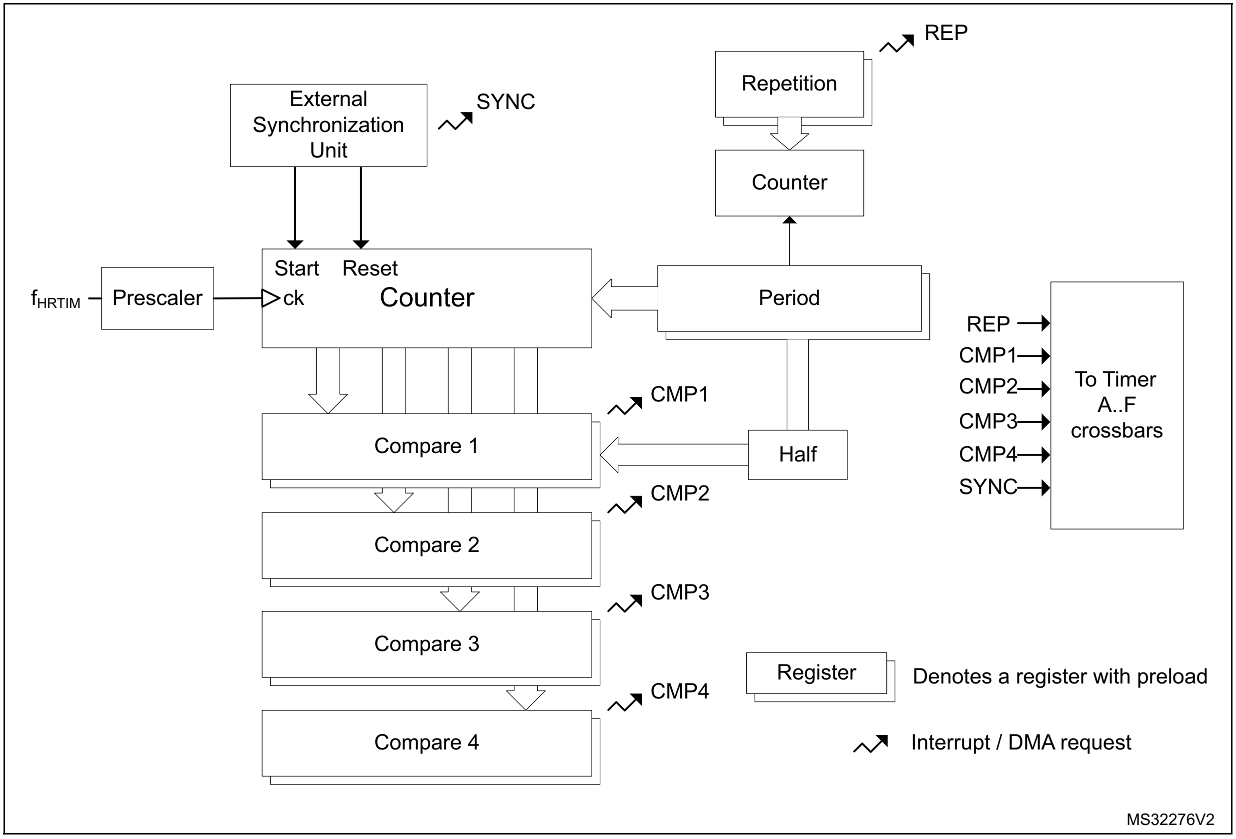
For the particular case where DTEN must be set while the burst mode is enabled with a deadtime upon entry (BME = 1, DIDL = 1, IDLEM = 1), it is necessary to force the two outputs in their IDLES state by software commands (SST, RST bits) before setting DTEN bit. This is to avoid any side effect resulting from a burst mode entry that would happen immediately before a deadtime enable.

27.3.5 Master timer

The main purpose of the master timer is to provide common signals to the 6 timing units, either for synchronization purpose or to set/reset outputs. It does not have direct control over any outputs, but still can be used indirectly by the set/reset crossbars.

Figure 204 provides an overview of the master timer.

Figure 204. Master timer overview



The master timer is based on the very same architecture as the timing units, with the following differences:

- It does not have outputs associated with, nor output related control
- It does not have its own crossbar unit, nor push-pull or deadtime mode
- It can only be reset by the external synchronization circuitry
- It does not have a capture unit, nor the auto-delayed mode
- It does not include external event blanking and windowing circuitry
- It has a limited set of interrupt / DMA requests: compare 1..4, repetition, register update and external synchronization event.

The master timer control register includes all the timer enable bits, for the master and timer A..F timing units. This allows to have all timer synchronously started with a single write access.

It also handles the external synchronization for the whole HRTIM timer (see *Section 27.3.19: Synchronizing the HRTIM with other timers or HRTIM instances*), with both MCU internal and external (inputs/outputs) resources.

Master timer control registers are mapped with the same offset as the timing units’ registers.

27.3.6 Up-down counting mode

The HRTIM is natively designed with up-counters. It offers however an operating mode with up-down counters, also called center-aligned mode.

This mode is enabled using the UDM bit in the HRTIM_TIMxCR2 register. This bit must not be changed once the timer is operating (TxEN bit set). It is only available for the TIMA..F. The master timer only works in up-counting mode.

Not all HRTIM features are supported in up-down counting. This section details the functional differences versus up-counting mode.

The period in HRTIM_PERxR must be preloaded (or static) in up-down mode. It can be updated only on period event or on counter reset.

The set/reset crossbar programming differs as follows:

The events coming from the timing units are setting/resetting the outputs depending on the counter up/down direction:

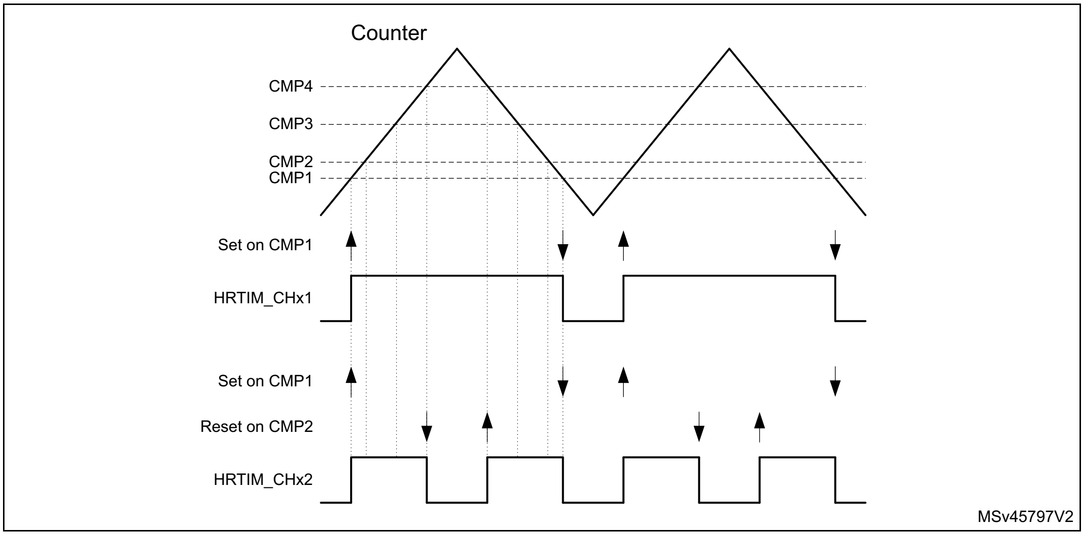
- If the event is enabled in the HRTIM_SETxyR register, it sets the output during up-counting and reset it during down-counting.
- If the event is enabled in the HRTIM_RSTxyR register, it resets the output during up-counting and set it during down-counting.
- If the events are enabled both in HRTIM_SETxyR and HRTIM_RSTxyR registers, the output toggles.

This applies to :

- – The timing unit: period, compare 1..4, register update (6 events)
- – The master timer: period, compare 1..4, HRTIM synchronization (6 events)
- – All other timing units (e.g. timer B..F for timer A): TIMEVNT1..9 (9 events described in *Table 216*)

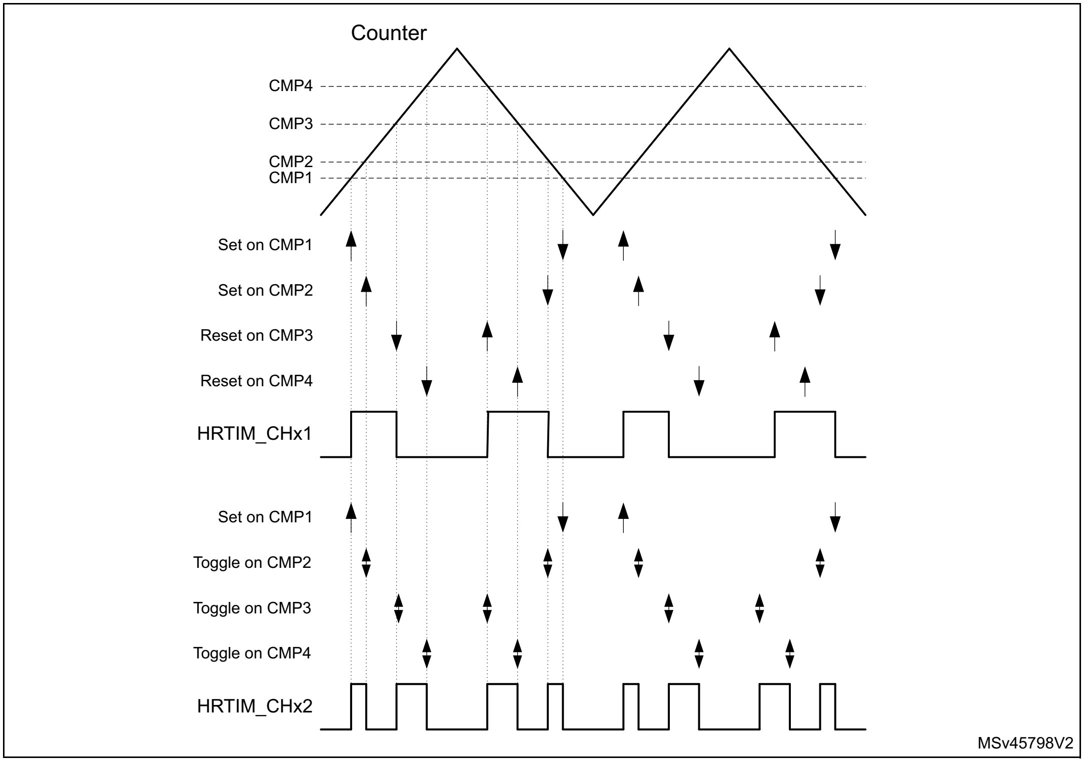
The *Figure 205* below shows how to generate basic waveforms.

Figure 205. Basic symmetric waveform in up-down counting mode



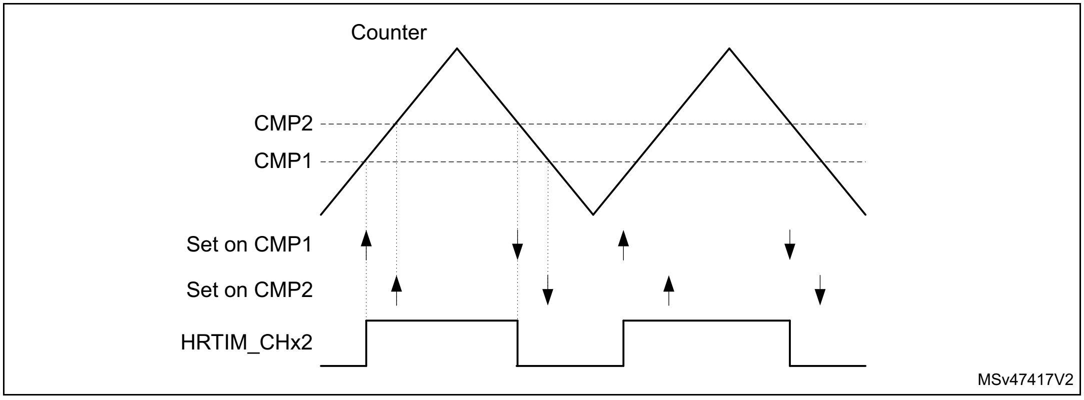
The *Figure 206* below shows how to generate some more complex waveforms, using the 4 available compare units and the toggle mode.

Figure 206. Complex symmetric waveform in up-down counting mode



The *Figure 207* shows how to generate an asymmetric waveform. In this case, it must be noticed that it is necessary to have compare 2 value greater than compare 1 value for the waveform to be asymmetric.

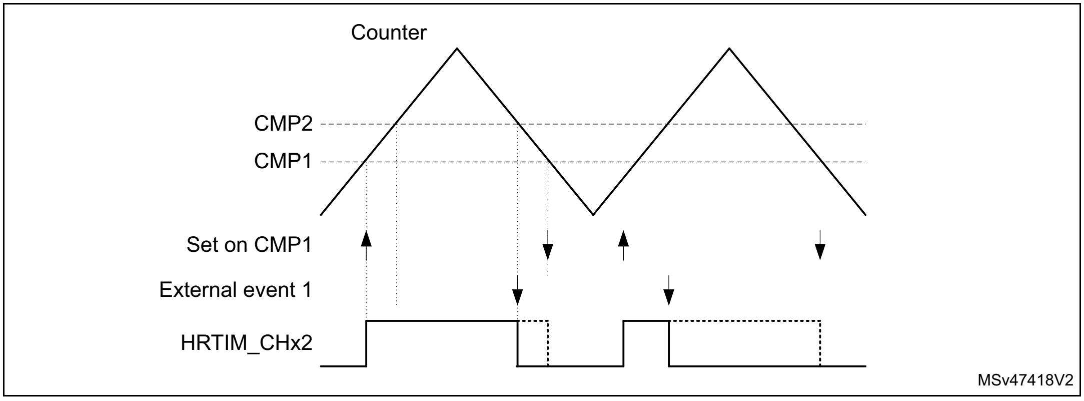
Figure 207. Asymmetric waveform in up-down counting mode



Note: *For asymmetric operation, it is required that CMP2 > CMP1.*

The behavior of the software forcing bits and external events EXTEVNT1..10 is identical for up-only and up-down counting mode. The *Figure 208* below shows how a pulse can be shorten in response to external events.

Figure 208. External event management in up-down counting mode



The up-down counting mode is available for both continuous and single-shot (retriggerable and non-retriggerable) operating modes. A reset causes the counter to re-start from 0. The *Figure 209* below shows the counter behavior in single-shot retriggerable mode, for TimB.

Figure 209. Interleaved up-down counter operation

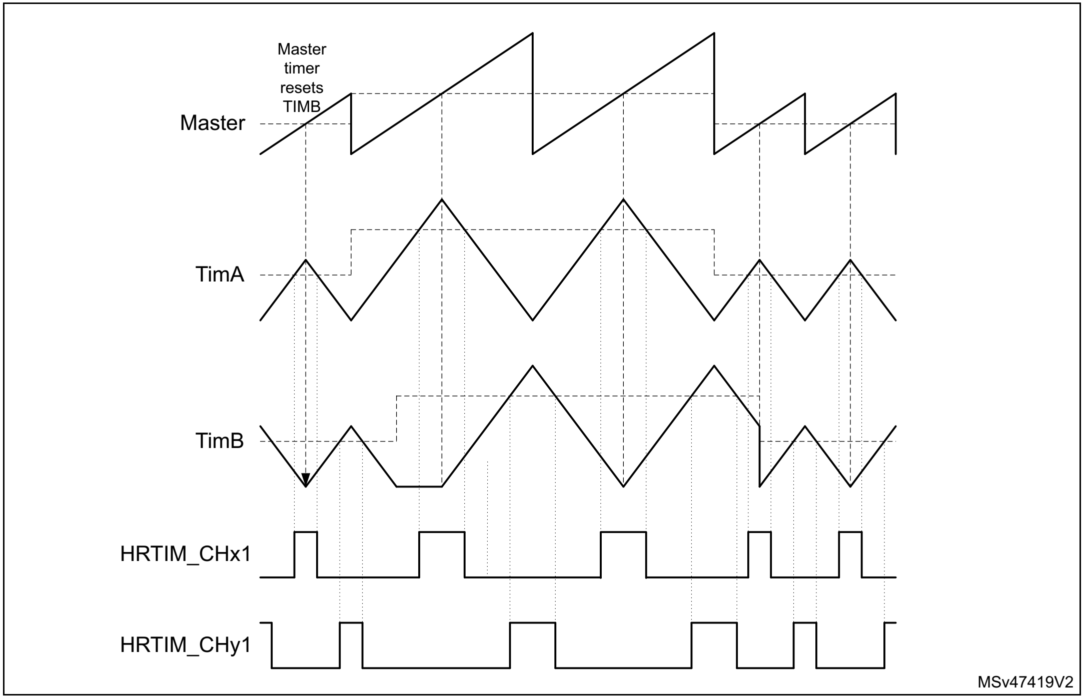
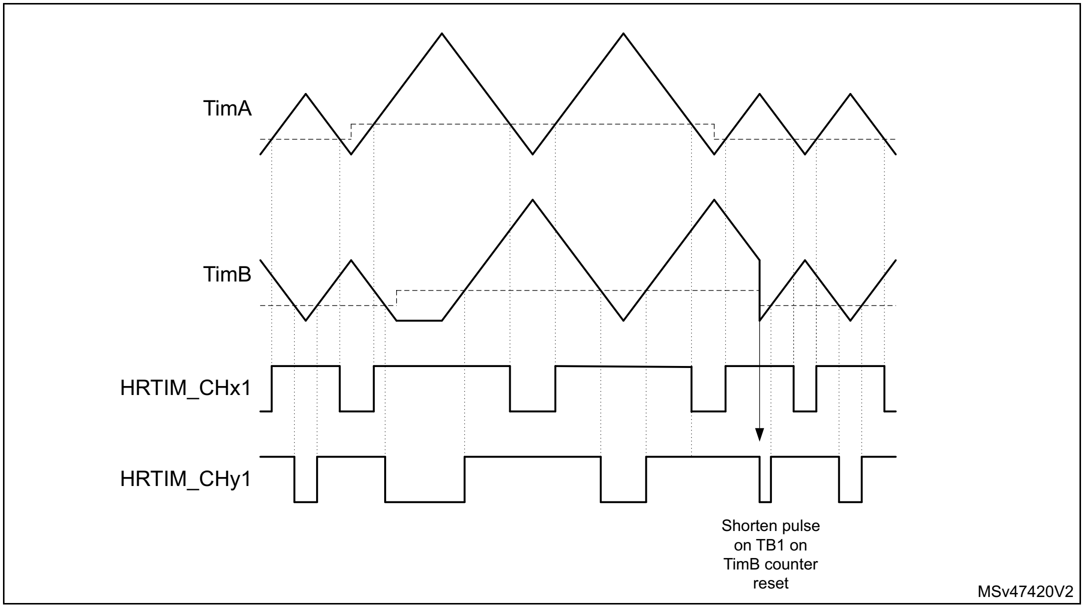


Figure 210. Interleaved up-down counter operation



Note: In up-down counting mode, the compare values must be 3 periods of the fHRTIM clock below the period value (TIMx_PER - 0xC0 if CKPSC[2:0] = 0, TIMx_PER - 0x60 if CKPSC[2:0] = 1, TIMx_PER - 0x30 if CKPSC[2:0] = 2,...). This applies for the compare events generated inside the timing unit. For compare events generated in other timing units, it is mandatory to avoid any event occurring within less than 1 period of the fHRTIM clock of a counter direction change (counter at 0, period event or counter reset).

The following features are supported in up-down counting mode:

- Half mode
- Deadtime insertion
- Push-pull mode, alternance push-pull done on when counter = 0 (see *Figure 211*).
- Delayed Idle
- Burst mode
- PWM mode with “greater than” comparison (see *Figure 212*).

Figure 211. Push-pull up-down mode example

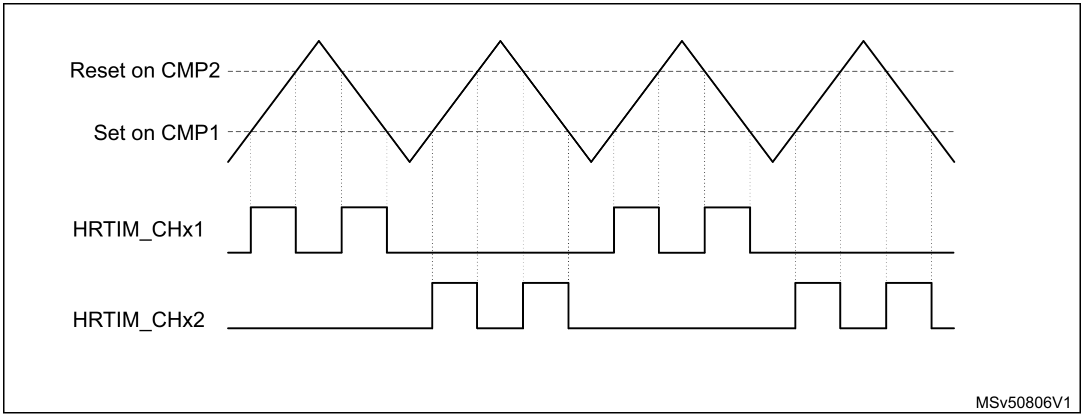
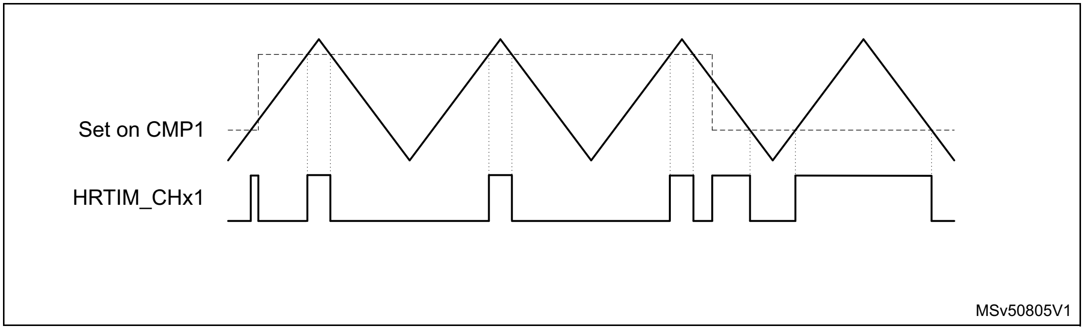


Figure 212. Up-down mode with “greater than” comparison



Caution: The following features are not supported in up-down counting mode:

- Auto-delayed mode
- Balanced Idle
- Triggered-half mode

The capture function is supported with the following differences:

- the capture value is referred to counting start, when up-counting
- the capture value is referred to the PER event when down counting
- the bit 16 of the capture register holds the counting direction status

The counter roll-over event is defined differently in-up-down counting mode to support various operating condition. It can be either generated:

- when the counter is equal to 0 (“valley” mode)
- when the counter reaches the period value set in the HRTIM_PERxR (“crest” mode)
- when both conditions are met (either 0 or HRTIM_PERxR value)

This event is used for multiple purposes in the HRTIM. The generation mode (valley, crest or both) can be programmed individually depending on the destination. *Table 220* summarizes the use cases and associated roll-over mode (xxROM[1:0]) programming bits in the HRTIM_TIMxCR2 register.

Table 220. Roll-over event destination and mode programming

Roll-over event use	Programming bits
Output set/reset	OUTROM[1:0]
Register content update trigger (transfer from preload to active)	ROM[1:0]
IRQ and/or DMA request trigger	ROM[1:0]
Burst mode clock source and /or burst start trigger	BMROM[1:0]
ADC trigger (see *Section : ADC post-scaler* for details)	ADROM[1:0]
External event filtering	ROM[1:0]
Repetition counter decrement	ROM[1:0]
Fault and event counter	FEROM[1:0]

Note: For events where both reset and roll-over are considered (IRQ/DMA and TxRSTU), the ROM[1:0] only influences the roll-over generation. The reset event is always taken into account whatever the ROM[1:0] value.

The roll-over event generation is defined as per following xxROM[1:0] bitfield setting:

- xxROM[1:0] = 00: event generated when both conditions are met (either 0 or HRTIM_PERxR value)
- xxROM[1:0] = 01: event generated when the counter is equal to 0 (“valley” mode) or when the counter is reset
- xxROM[1:0] = 10: event generated when the counter reaches the period value set in the HRTIM_PERxR (“crest” mode)

Figure 213 here-below shows a push-pull up-down mode with set on period event and OUTROM[1:0] =10.

Figure 213. Up-down mode with output set on period event, for OUTROM[1:0]=10

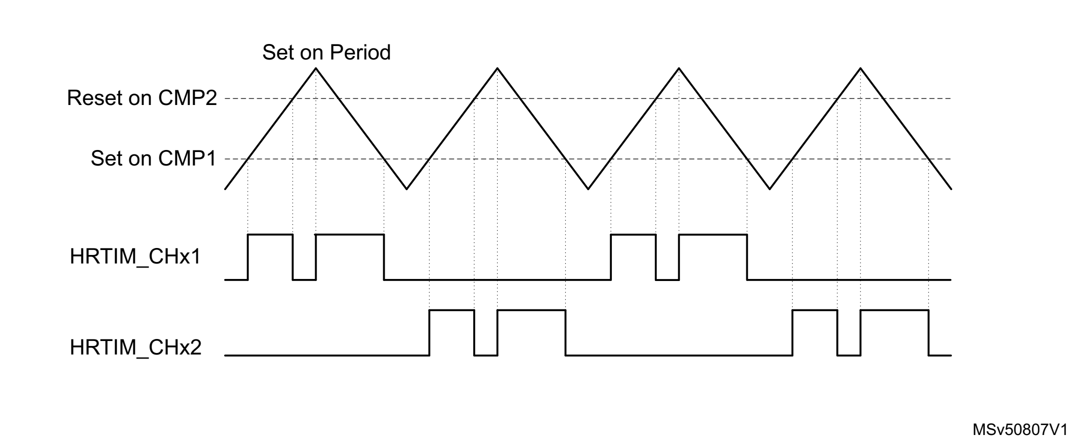
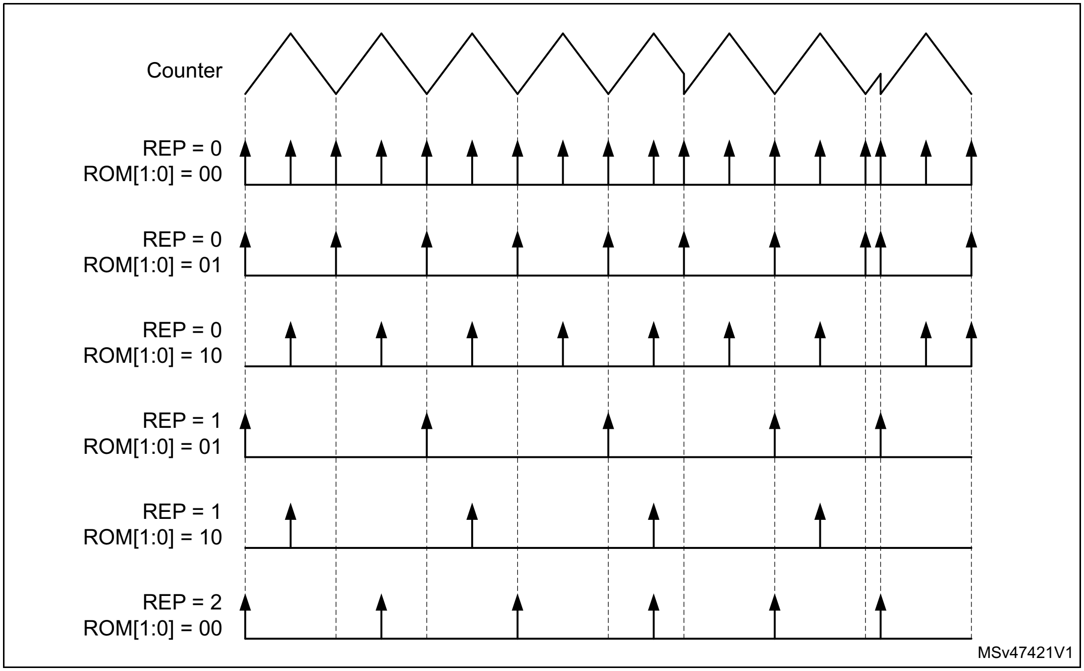


Figure 214 below shows how the repetition counter is decremented in up-down counting mode.

Figure 214. Repetition counter behavior in up-down counting mode



The dual channel DAC triggers are working as for the up-counting mode.

The event blanking and windowing differs so as to have the blanking or windowing done within the output pulse, at a programmable time. The EExFLTR[3:0] codes are depending on the UDM bit setting, as per the *Table 221* below. Whenever the roll-over event is used for blanking or windowing, the ROM[1:0] programming applies for defining when it is generated.

Table 221. EExFLTR[3:0] codes depending on UDM bit setting

EE1FLTR[3:0]	Up-counting mode (UDM = 0)	Up/down-counting mode (UDM = 1)
0010	Blanking from counter reset/roll-over to compare 2	Blanking from compare 1 to compare 2, only during the up-counting phase
0100	Blanking from counter reset/roll-over to compare 4	Blanking from compare 3 to compare 4, only during the up-counting phase
1101	Windowing from counter reset/roll-over to compare 2	Windowing from compare 2 to compare 3, only during the up-counting phase
1110	Windowing from counter reset/roll-over to compare 3	Windowing from compare 2 to compare 3, only during the down-counting phase
1111	Windowing from another timing unit: TIMWIN source (see *Table 225* for details)	Windowing from compare 2 during the up-counting phase to compare 3 during the down-counting phase

27.3.7 Set / reset events priorities and narrow pulses management

This section describes how the output waveform is generated when several set and/or reset requests are occurring within 3 consecutive t_{HRTIM} periods.

Case 1: clock prescaler CKPSC[2:0] < 5

An arbitration is performed during each t_{HRTIM} period, in 3 steps:

1. For each active event, the desired output transition is determined (set, reset or toggle).
2. A predefined arbitration is performed among the active events (from highest to lowest priority CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1 ⟶ PER, see *Section : Concurrent set requests/ Concurrent reset requests*).
3. A high-resolution delay-based arbitration is performed with reset having the highest priority, among the low-resolution events and events having won the predefined arbitration.

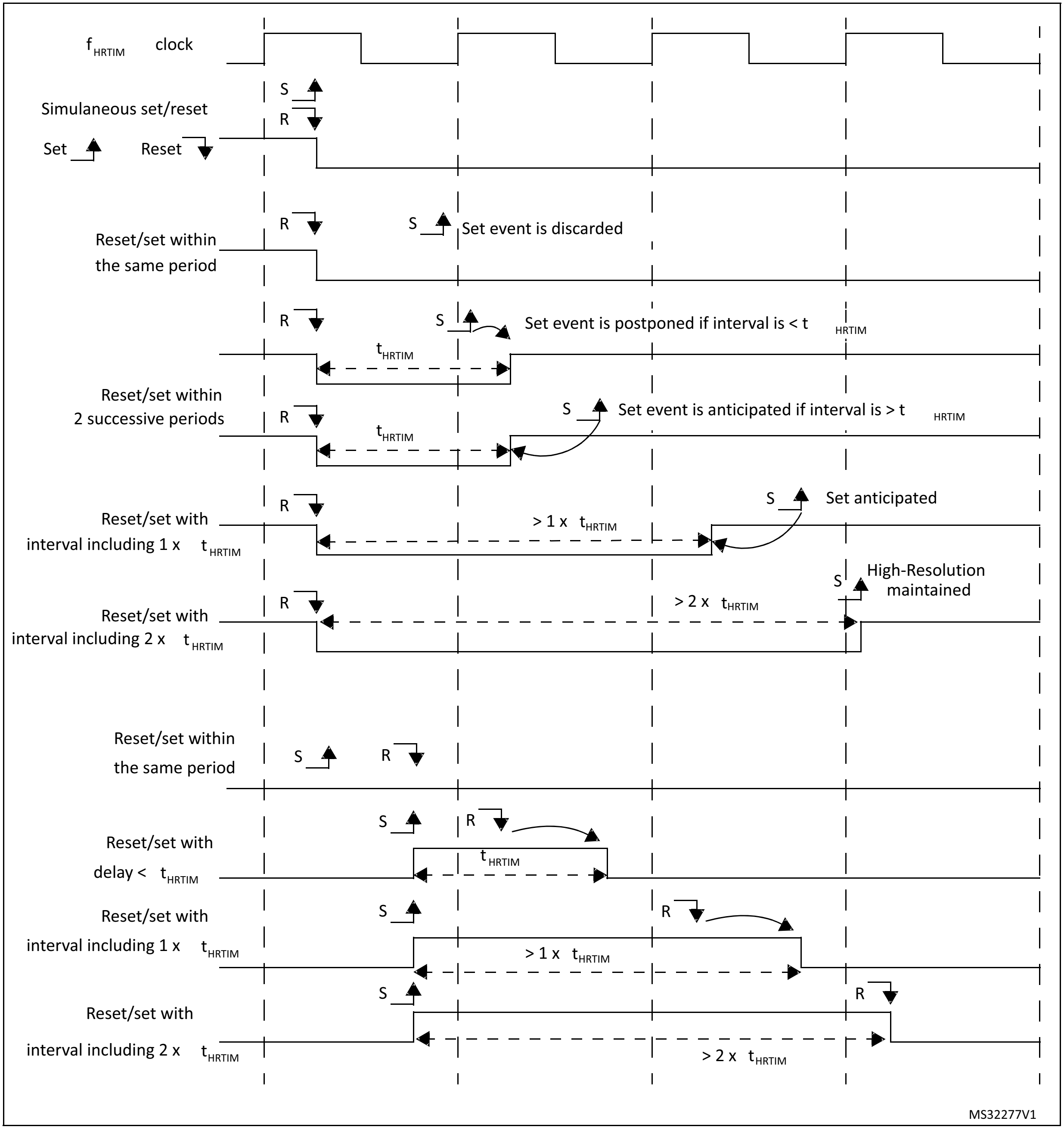
When set and reset requests from two different sources are simultaneous, the reset action has the highest priority. If the interval between set and reset requests is below 2 f_{HRTIM} period, the behavior depends on the time interval and on the alignment with the f_{HRTIM} clock, as shown on *Figure 215*.

Note: If the set and reset requests are simultaneous and coming from the same timing unit, the CMPx priority applies, as shown in step 2 here-above. For instance, taking CMP2 = CMP4:

- If CMP2 does a set and CMP4 a reset, the output is reset.

- If CMP2 does a reset and CMP4 a set, the output is set.

Figure 215. Short distance set/reset management for narrow pulse generation


fHRTIM clock
Simulaneous set/reset
Set
Reset
S
R
Reset/set within the same period
Set event is discarded
Set event is postponed if interval is < tHRTIM
tHRTIM
Reset/set within 2 successive periods
Set event is anticipated if interval is > tHRTIM
Reset/set with interval including 1 x tHRTIM
> 1 x tHRTIM
Set anticipated
Reset/set with interval including 2 x tHRTIM
> 2 x tHRTIM
High-Resolution maintained
Reset/set within the same period
Reset/set with delay < tHRTIM
Reset/set with interval including 1 x tHRTIM
Reset/set with interval including 2 x tHRTIM
MS32277V1

If the set and reset events are generated within the same t_{HRTIM} period, the reset event has the highest priority and the set event is ignored.

If the set and reset events are generated with an interval below t_{HRTIM} period, across 2 periods, a pulse of 1 t_{HRTIM} period is generated.

If the set and reset events are generated with an interval below 2 t_{HRTIM} periods, a pulse of 2 t_{HRTIM} periods is generated.

If the set and reset events are generated with an interval between 2 and 3 t_{HRTIM} periods, the high-resolution is available if the interval is over 2 complete t_{HRTIM} periods.

If the set and reset events are generated with an interval above 3 t_{HRTIM} periods, the high-resolution is always available.

Concurrent set requests/ Concurrent reset requests

When multiple sources are selected for a set event, an arbitration is performed when the set requests occur within the same f_{HRTIM} clock period.

In case of multiple requests from adjacent timers (TIMEVNT1..9), the request which occurs first is taken into account. The arbitration is done in 2 steps, depending on:

1. the source (CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1),
2. the delay.

If multiple requests from the master timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and a single request is taken into account, whatever the effective high-resolution setting (from the highest to the lowest priority):

MSTCMP4 → MSTCMP3 → MSTCMP2 → MSTCMP1 → MSTCMPER

Note: It is advised to avoid generating multiple set (reset) requests from the master timer to a given timer with an interval below 3x t_{HRTIM} *to maintain the high-resolution.*

When multiple requests internal to the timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and the requests are taken with the following priority, whatever the effective timing (from highest to lowest):

CMP4 → CMP3 → CMP2 → CMP1 → PER

Note: Practically, this is of a primary importance when multiple compare events can be simultaneously generated or when using auto-delayed compare 2 and compare 4 simultaneously (i.e. when the effective set/reset cannot be determined a priori because it is related to an external event). In this case, the highest priority signal must be affected to the CMP4 event.

Last, the highest priority is given to low-resolution events: EXTEVNT1..10, RESYNC (coming from SYNC event if SYNCRSTx or SYNCSTRTx is set or from a software reset), update and software set (SST). The update event is considered as having the largest delay (0x1F if PSC = 0).

As a summary, in case of a close vicinity (events occurring within the same f_{HRTIM} clock period), the effective set (reset) event is arbitrated between:

- Any TIMEVNT1..9 event
- A single source from the master (as per the fixed arbitration given above)
- A single source from the timer
- The “low-resolution events”.

The same arbitration principle applies for concurrent reset requests. In this case, the reset request has the highest priority.

Case 2: clock prescaler CKPSC[2:0] ≥ 5

The narrow pulse management is simplified when the high-resolution is not effective.

A set or reset event occurring within the prescaler clock cycle is delayed to the next active edge of the prescaled clock (as for a counter reset), even if the arbitration is still performed every t_{HRTIM} cycle.

If a reset event is followed by a set event within the same prescaler clock cycle, the latest event is considered.

27.3.8 External events global conditioning

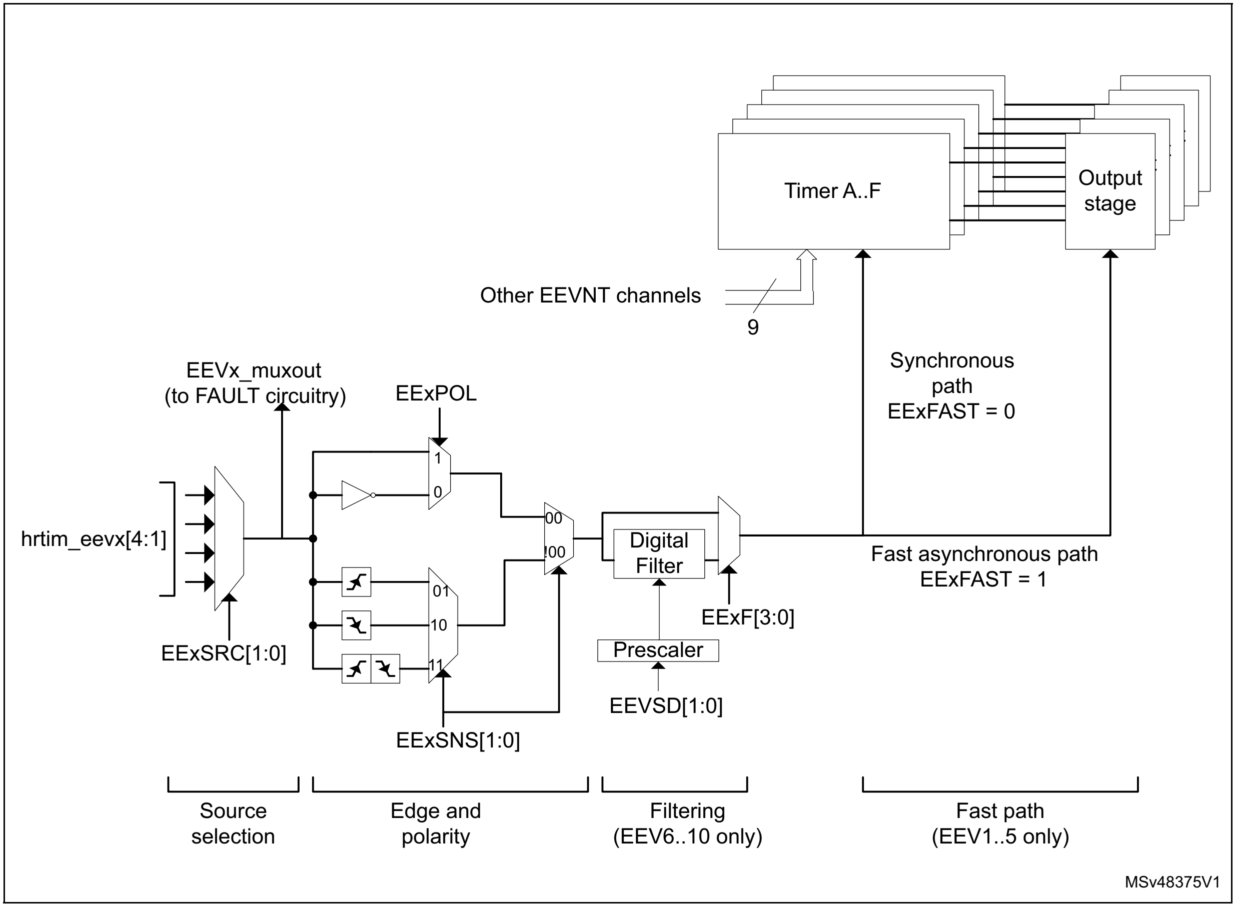
The HRTIM timer can handle events not generated within the timer, referred to as “external event”. These external events come from multiple sources, either on-chip or off-chip:

- built-in comparators,
- digital input pins (typically connected to off-chip comparators and zero-crossing detectors),
- on-chip events for other peripheral (ADC’s analog watchdogs and general purpose timer trigger outputs).

The external events conditioning circuitry allows to select the signal source for a given channel (with a 4:1 multiplexer) and to convert it into an information that can be processed by the crossbar unit (for instance, to have an output reset triggered by a falling edge detection on an external event channel).

Up to 10 external event channels can be conditioned and are available simultaneously for any of the 6 timers. This conditioning is common to all timers, since this is usually dictated by external components (such as a zero-crossing detector) and environmental conditions (typically the filter set-up is related to the applications noise level and signature). *Figure 216* presents an overview of the conditioning logic for a single channel.

Figure 216. External event conditioning overview (1 channel represented)



The 10 external events are initialized using the HRTIM_EECR1 and HRTIM EECR2 registers:

- to select up to 4 sources with the EExSRC[1:0] bits,
- to select the sensitivity with EExSNS[1:0] bits, to be either level-sensitive or edge-sensitive (rising, falling or both),
- to select the polarity, in case of a level sensitivity, with EExPOL bit,
- to have a low latency mode, with EExFAST bits (see *Section : Latency to external events*), for external events 1 to 5.

Note: The external events used as triggers for reset, capture, burst mode, ADC triggers and delayed protection are edge-sensitive even if EESNS bit is reset (level-sensitive selection): if POL = 0 the trigger is active on external event rising edge, while if POL = 1 the trigger is active on external event falling edge.

The external events are discarded as long as the counters are disabled (TxCEN bit reset) to prevent any output state change and counter reset, except if they are used as ADC triggers.

Additionally, it is possible to enable digital noise filters, for external events 6 to 10, using EExF[3:0] bits in the HRTIM_EECR3 register.

A digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected).

The sampling clock is either the f_{HRTIM} clock or a specific prescaled clock f_{EEVS} derived from f_{HRTIM}, defined with EEVSD[1:0] bits in HRTIM_EECR3 register.

Table 222 summarizes the available features associated with each of the 10 external events channels.The features and sources are summarized in *Table 208*.

Table 222. External events features

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E,F
hrtim_eev1[4:1]	Yes	-	-	-
hrtim_eev2[4:1]	Yes	-	-	-
hrtim_eev3[4:1]	Yes	-	-	-
hrtim_eev4[4:1]	Yes	-	-	-
hrtim_eev5[4:1]	Yes	-	-	-
hrtim_eev6[4:1]	-	Yes	Yes	-
hrtim_eev7[4:1]	-	Yes	Yes	-
hrtim_eev8[4:1]	-	Yes	-	Yes
hrtim_eev9[4:1]	-	Yes	-	Yes
hrtim_eev10[4:1]	-	Yes	-	-

Latency to external events

The external event conditioning gives the possibility to adjust the external event processing time (and associated latency) depending on performance expectations:

- A regular operating mode, in which the external event is resampled with the clock before acting on the output crossbar. This adds some latency but gives access to all crossbar functionalities. It enables the generation of an externally triggered high-resolution pulse.
- A fast operating mode, in which the latency between the external event and the action on the output is minimized. This mode is convenient for ultra-fast over-current protections, for instance.

EExFAST bits in the HRTIM_EECR1 register allow to define the operating for channels 1 to 5. This influences the latency and the jitter present on the output pulses, as summarized in the table below.

Table 223. Output set/reset latency and jitter versus external event operating mode

EExFAST	Response time latency	Response time jitter	Jitter on output pulse (counter reset by ext. event)
0	5 to 6 cycles of f_{HRTIM} clock	1 cycles of f_{HRTIM} clock	No jitter, pulse width maintained with high-resolution
1	Minimal latency (depends whether the comparator or digital input is used)	Minimal jitter	1 cycle of f_{HRTIM} clock jitter pulse width resolution down to t_{HRTIM}

The EExFAST mode is only available with level-sensitive programming (EExSNS[1:0] = 00); the edge-sensitivity cannot be programmed.

It is possible to apply event filtering to external events (both blanking and windowing with EExFLTR[3:0] != 0000, see *Section 27.3.9*). In this case, EExLTCHx bit must be reset: the postponed mode is not supported, neither the windowing timeout feature.

Note: The external event configuration (source and polarity) must not be modified once the related EExFAST bit is set.

A fast external event cannot be used to toggle an output: if must be enabled either in HRTIM_SETxyR or HRTIM_RSTxyR registers, not in both.

When a set and a reset event - from 2 independent fast external events - occur simultaneously, the reset has the highest priority in the crossbar and the output becomes inactive.

When EExFAST bit is set, the output cannot be changed during the 11 f_{HRTIM} clock periods following the external event.

Figure 217 and *Figure 218* give practical examples of the reaction time to external events, for output set/reset and counter reset.

Figure 217. Latency to external events falling edge (counter reset and output set)

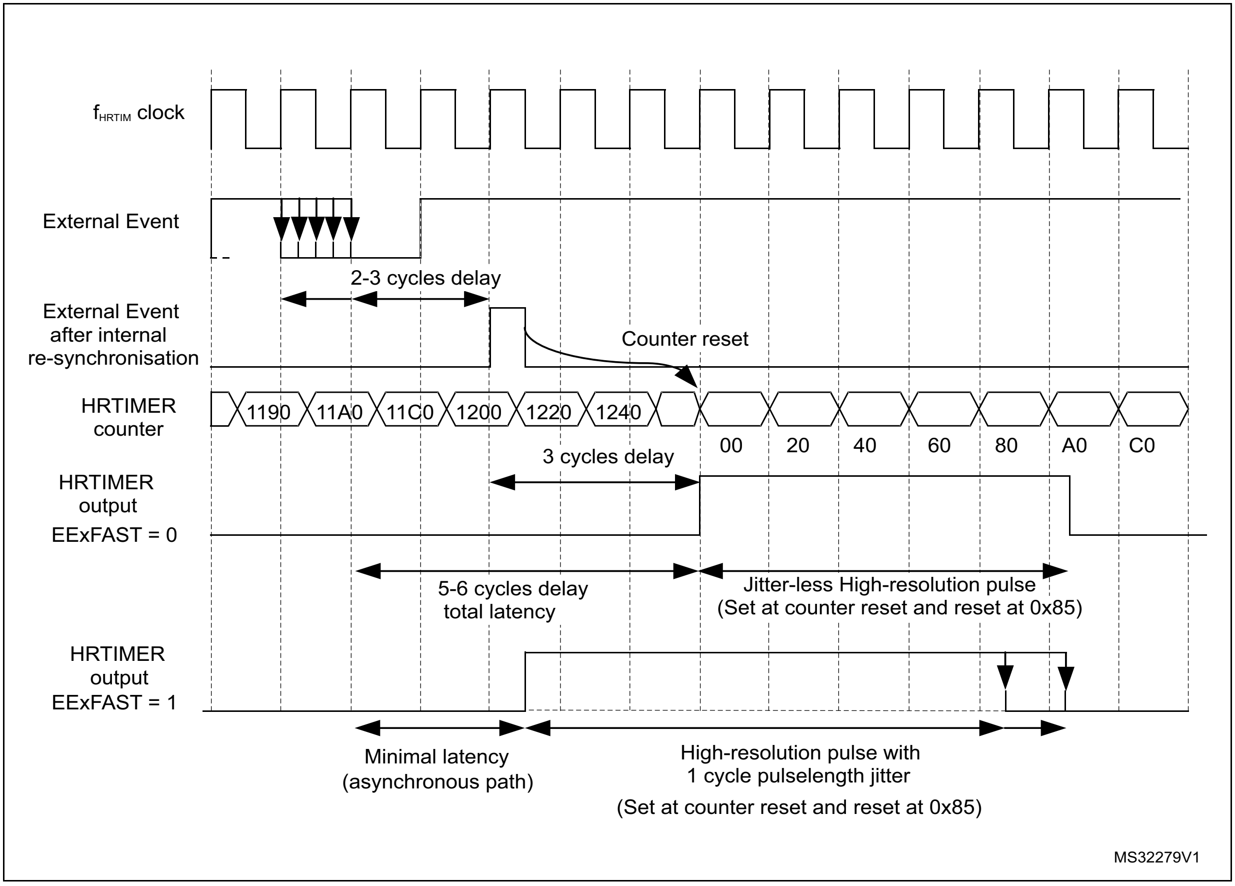
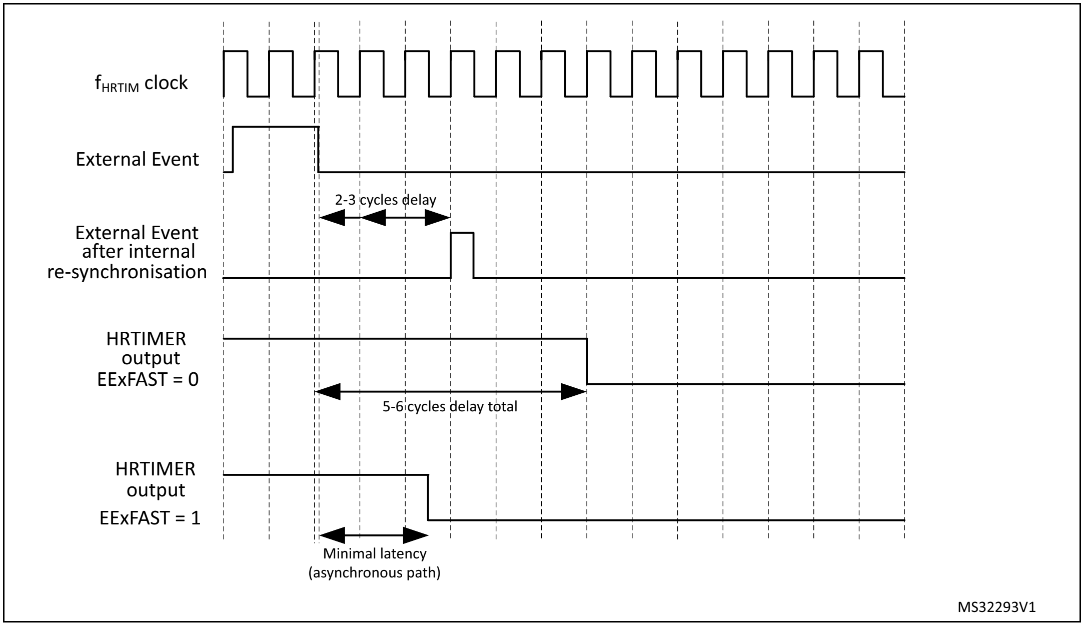

f_{HRTIM} clock
External Event
2-3 cycles delay
External Event after internal re-synchronisation
Counter reset
HRTIMER counter
1190
11A0
11C0
1200
1220
1240
00
20
40
60
80
A0
C0
3 cycles delay
HRTIMER output EExFAST = 0
5-6 cycles delay total latency
Jitter-less High-resolution pulse (Set at counter reset and reset at 0x85)
HRTIMER output EExFAST = 1
Minimal latency (asynchronous path)
High-resolution pulse with 1 cycle pulselength jitter
(Set at counter reset and reset at 0x85)
MS32279V1

Figure 218. Latency to external events (output reset on external event)


f_{HRTIM} clock
External Event
2-3 cycles delay
External Event after internal re-synchronisation
HRTIMER output EExFAST = 0
5-6 cycles delay total
HRTIMER output EExFAST = 1
Minimal latency (asynchronous path)
MS32293V1

27.3.9 External event filtering in timing units

Once conditioned, the 10 external events are available for all timing units.

They can be used directly and are active as soon as the timing unit counter is enabled (TxCEN bit set).

They can also be filtered to have an action limited in time, usually related to the counting period. Two operations can be performed:

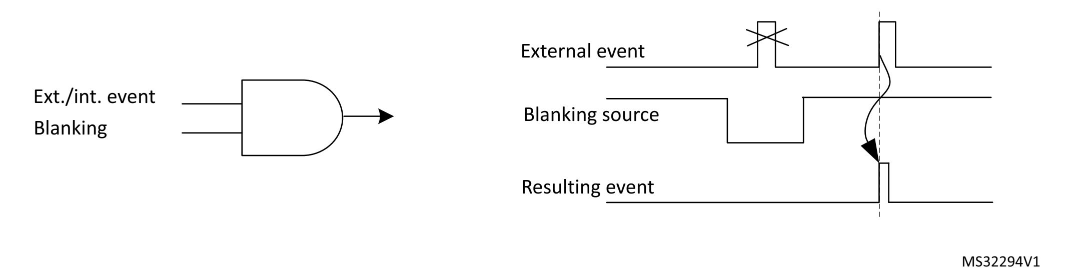
- blanking, to mask external events during a defined time period,
- windowing, to enable external events only during a defined time period.

These modes are enabled using HRTIM_EExFLTR[3:0] bits in the HRTIM_EEFxR1 and HRTIM_EEFxR2 registers. Each of the 6 timer A..F timing units has its own programmable filter settings for the 10 external events.

Blanking mode

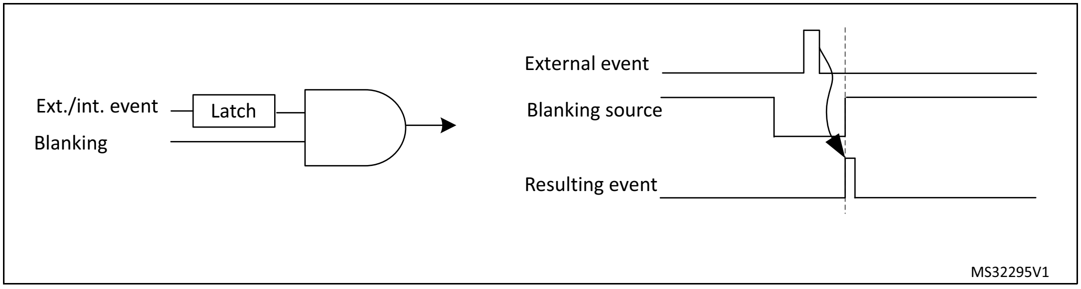
In event blanking mode (see *Figure 219*), the external event is ignored if it happens during a given blanking period. This is convenient, for instance, to avoid a current limit to trip on switching noise at the beginning of a PWM period. This mode is active for EExFLTR[3:0] bitfield values ranging from 0001 to 1100.

Figure 219. Event blanking mode



In event postpone mode, the external event is not taken into account immediately but is memorized (latched) and generated as soon as the blanking period is completed, as shown on *Figure 220*. This mode is enabled by setting EExLTCH bit in HRTIM_EEFxR1 and HRTIM_EEFxR2 registers.

Figure 220. Event postpone mode



The blanking signal comes from several sources:

- the timer itself: the blanking lasts from the counter reset to the compare match (EExFLTR[3:0] = 0001 to 0100 for compare 1 to compare 4). In up/down mode (UDM bit set to 1), the counter reset event is defined as per the ROM[1:0] bit setting.
- from other timing units (EExFLTR[3:0] = 0101 to 1100): the blanking lasts from the selected timing unit counter reset to one of its compare match, or can be fully programmed as a waveform on Tx2 output. In this case, events are masked as long as the Tx2 signal is inactive (it is not necessary to have the output enabled, the signal is taken prior to the output stage).

The EEXFLTR[3:0] configurations from 0101 to 1100 are referred to as TIMFLTR1 to TIMFLTR8 in the bit description, and differ from one timing unit to the other. *Table 224* gives the 8 available options per timer: CMPx refers to blanking from counter reset to compare match, Tx2 refers to the timing unit TIMx output 2 waveform defined with HRTIM_SETx2 and HRTIM_RSTx2 registers. For instance, timer B (TIMFLTR6) is timer C output 2 waveform.

Table 224. Filtering signals mapping per timer

Source		Timer A				Timer B				Timer C				Timer D				Timer E				Timer F			
		CMP1	CMP2	CMP4	TA2	CMP1	CMP2	CMP4	TB2	CMP1	CMP2	CMP4	TC2	CMP1	CMP2	CMP4	TD2	CMP1	CMP2	CMP4	TE2	CMP1	CMP2	CMP4	TF2
Destination	Timer A	-				1	-	2	3	4	-	5	-	7	-	-	-	-	8	-	-	6	-	-	-
	Timer B	1	-	2	3	-				4	5	-	-	-	7	-	-	8	-	-	-	-	6	-	-
	Timer C	-	1	-	-	2	-	3	-	-				5	-	6	7	-	-	8	-	4	-	-	-
	Timer D	1	-	-	-	-	2	-	-	3	4	-	5	-				6	-	7	-	-	-	8	-
	Timer E	-	1	-	-	2	-	-	-	3	-	-	-	6	-	7	8	-				-	-	4	5
	Timer F	-	-	1	-	-	2	-	-	-	-	3	-	-	4	5	-	6	-	7	8	-			

Figure 221 and *Figure 222* give an example of external event blanking for all edge and level sensitivities, in regular and postponed modes.

Figure 221. External trigger blanking with edge-sensitive trigger

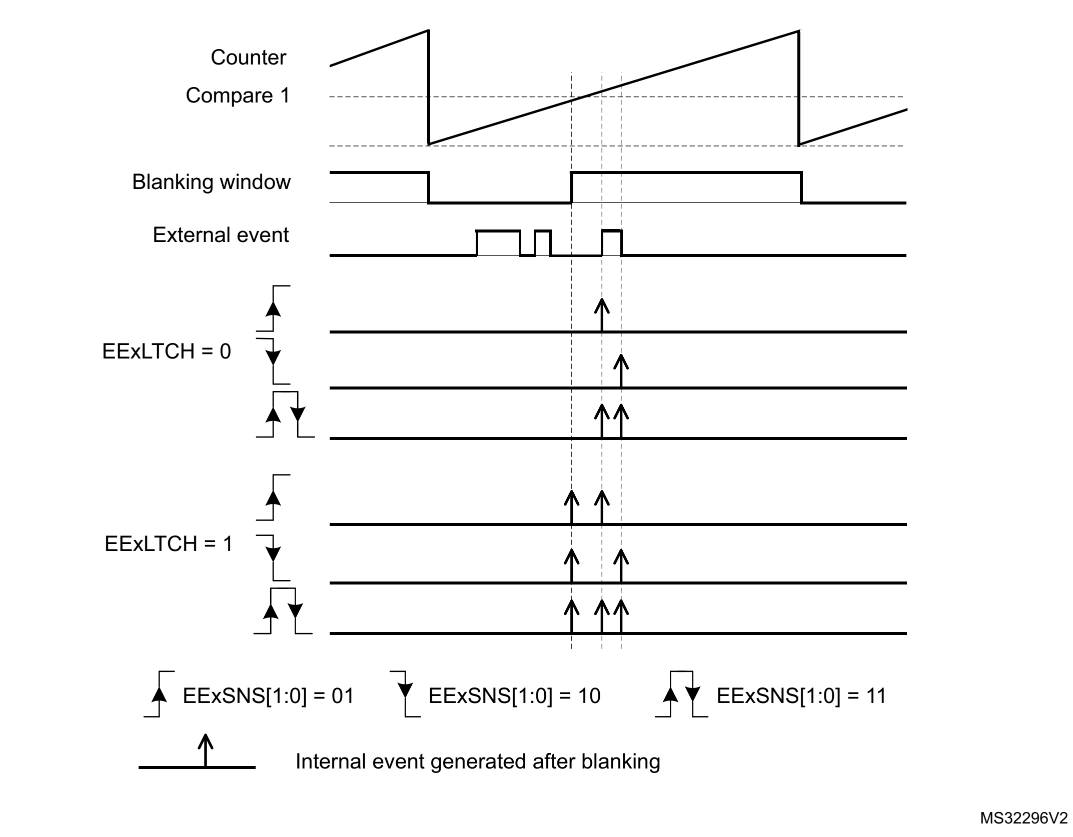
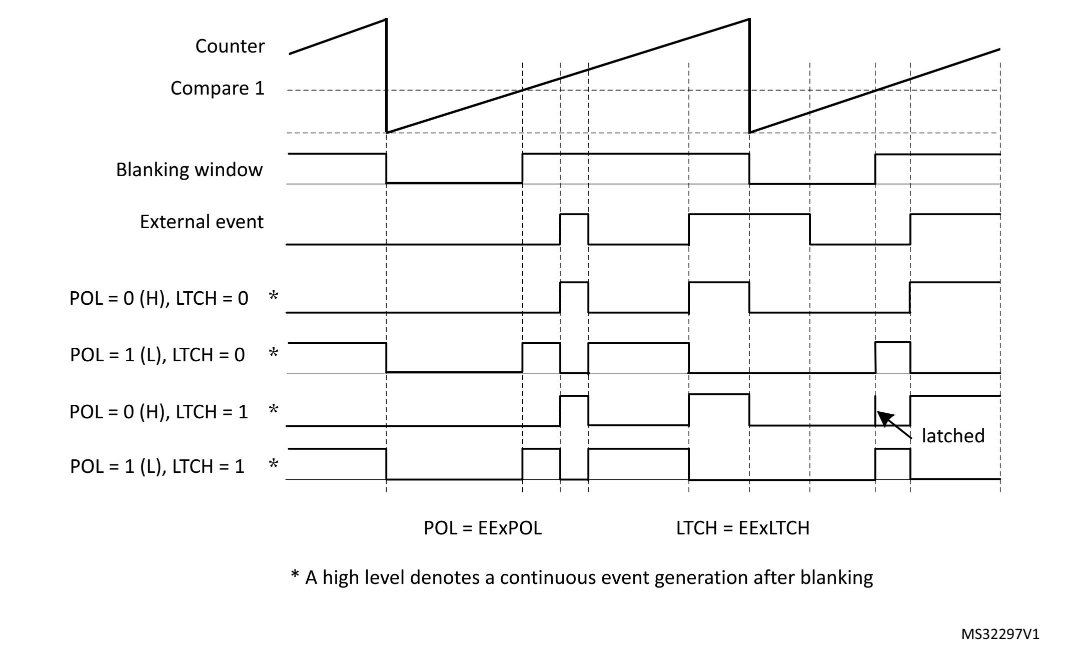


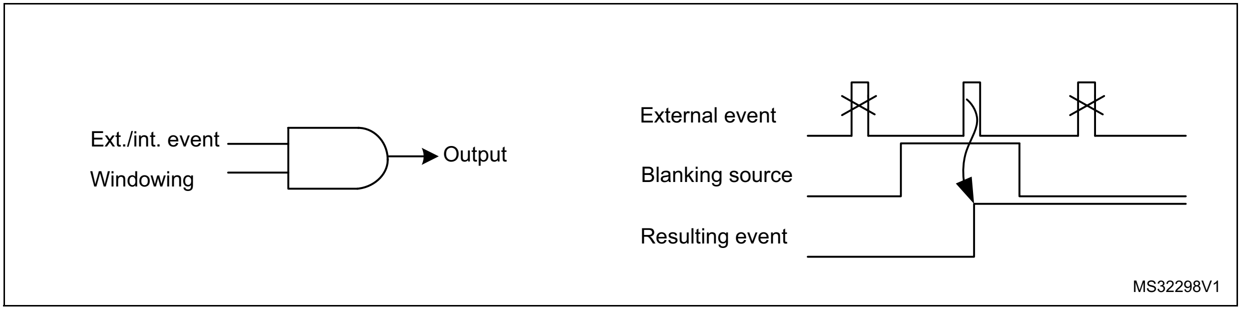
Figure 222. External trigger blanking, level sensitive triggering



Windowing mode

In event windowing mode, the event is taken into account only if it occurs within a given time window, otherwise it is ignored. This mode is active for EExFLTR[3:0] ranging from 1101 to 1111.

Figure 223. Event windowing mode



EExLTCH bit in EEFxR1 and EEFxR2 registers allows to latch the signal, if set to 1: in this case, an event is accepted if it occurs during the window but is delayed at the end of it.

- If EExLTCH bit is reset and the signal occurs during the window, it is passed through directly.
- If EExLTCH bit is reset and no signal occurs, a timeout event is generated at the end of the window.

A use case of the windowing mode is to filter synchronization signals. The timeout generation allows to force a default synchronization event, when the expected synchronization event is lacking (for instance during a converter start-up).

There are 3 sources for each external event windowing, coded as follows:

- 1101 and 1110: the windowing lasts from the counter reset to the compare match (respectively compare 2 and compare 3). In up/down mode (UDM bit set to 1), the counter reset event is defined as per the ROM[1:0] bit setting.
- 1111: the windowing is related to another timing unit and lasts from its counter reset to its compare 2 match. The source is described as TIMWIN in the bit description and is given in *Table 225*. As an example, the external events in timer B can be filtered by a window starting from timer A counter reset to timer A compare 2.

Table 225. Windowing signals mapping per timer (EEFLTR[3:0] = 1111)

Destination	Timer A	Timer B	Timer C	Timer D	Timer E	Timer F
TIMWIN (source)	Timer B CMP2	Timer A CMP2	Timer D CMP2	Timer C CMP2	Timer F CMP2	Timer E CMP2

Note: The timeout event generation is not supported if the external event is programmed in fast mode.

Figure 224 and *Figure 225* present how the events are generated for the various edge and level sensitivities, as well as depending on EExLTCH bit setting. Timeout events are specifically mentioned for clarity reasons.

Figure 224. External trigger windowing with edge-sensitive trigger

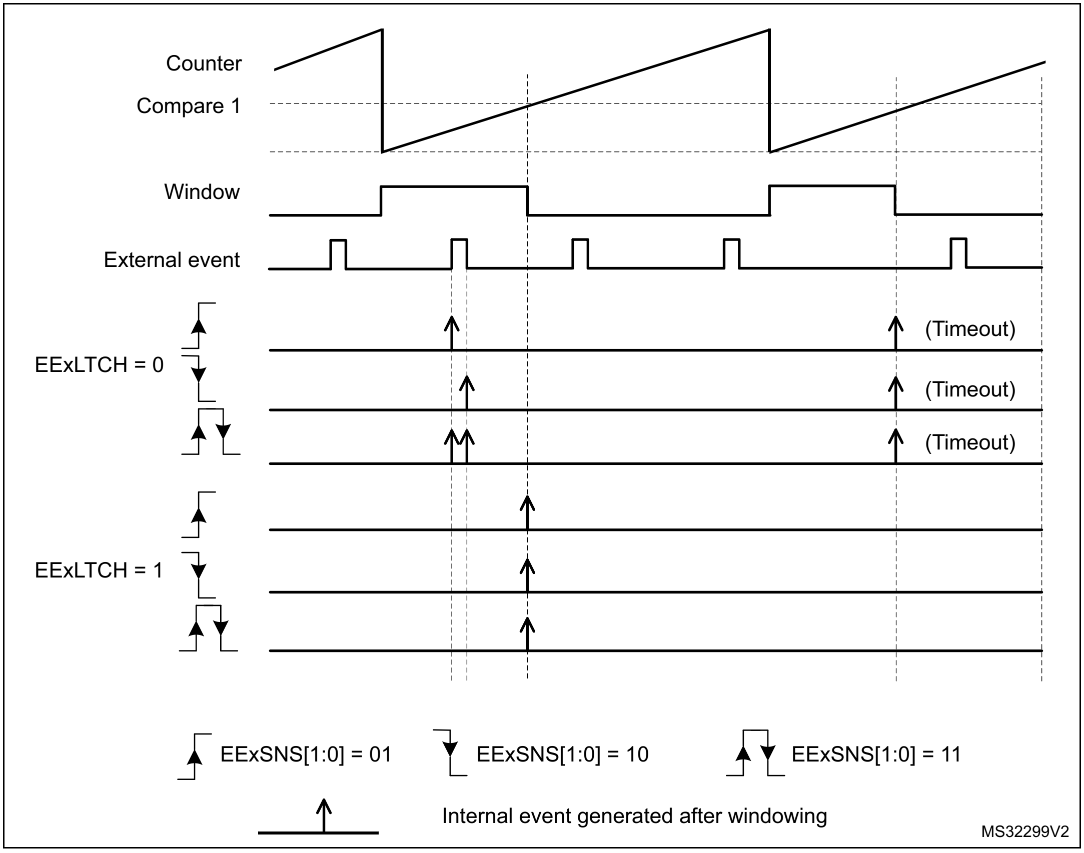
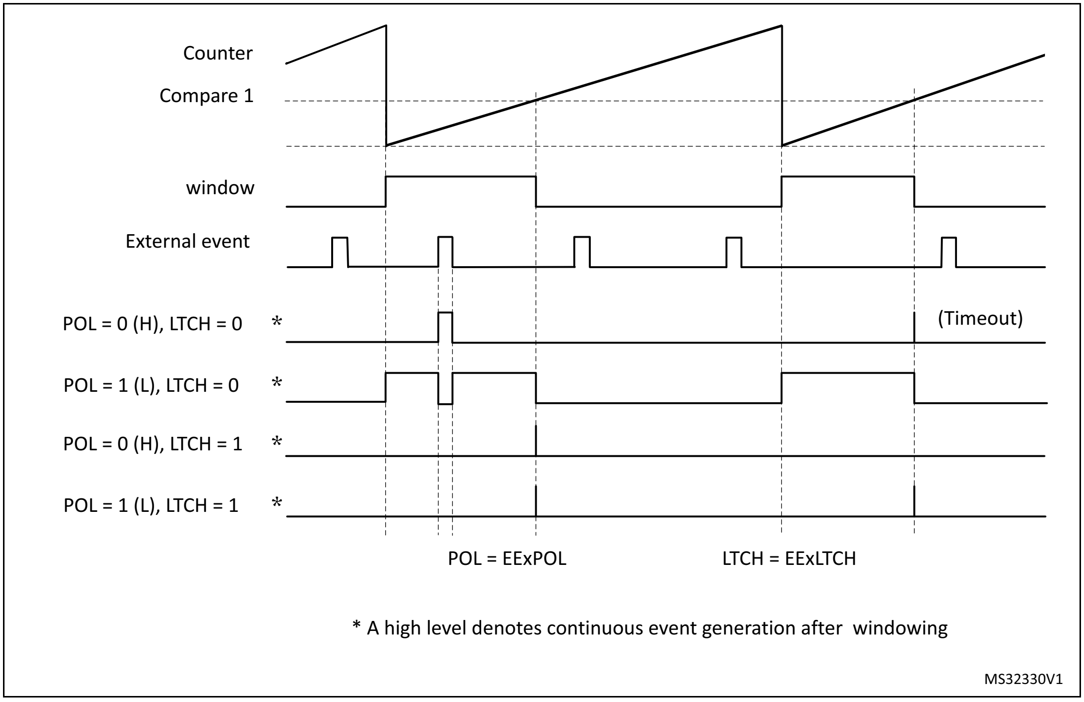


Figure 225. External trigger windowing, level sensitive triggering

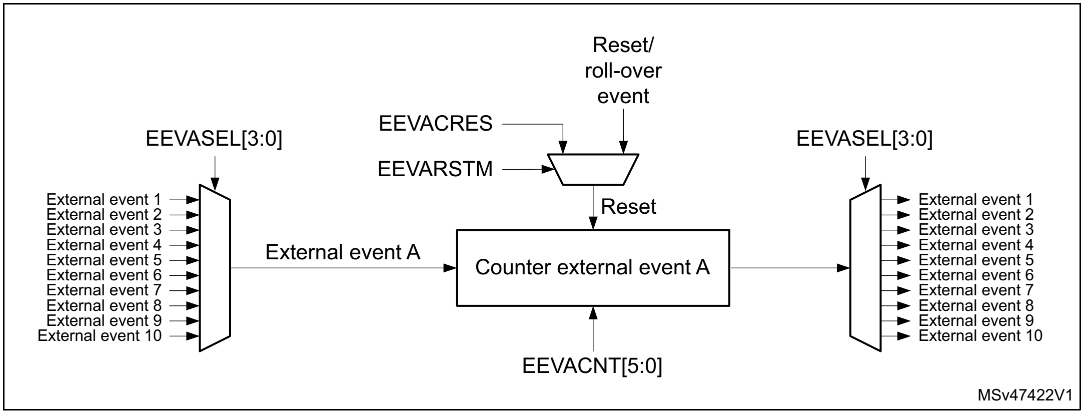


External event counter

Each timing unit also features an external event counter following the filtering unit, typically for valley skipping implementation.

The circuitry allows to filter any of the 10 external events filtered, as shown on *Figure 226*.

Figure 226. External event counter – channel A



The counter is enabled using the EEVACE bit in the HRTIM_EEFxR3 register. This mode is only valid for edge-sensitive external events (EEASNS[1:0] bit = 01,10 or 11).

The external event is propagated to the timer only if the number of active edges is greater or equal to the value programmed in (EEVACNT[5:0]+1).

Two operating modes are available:

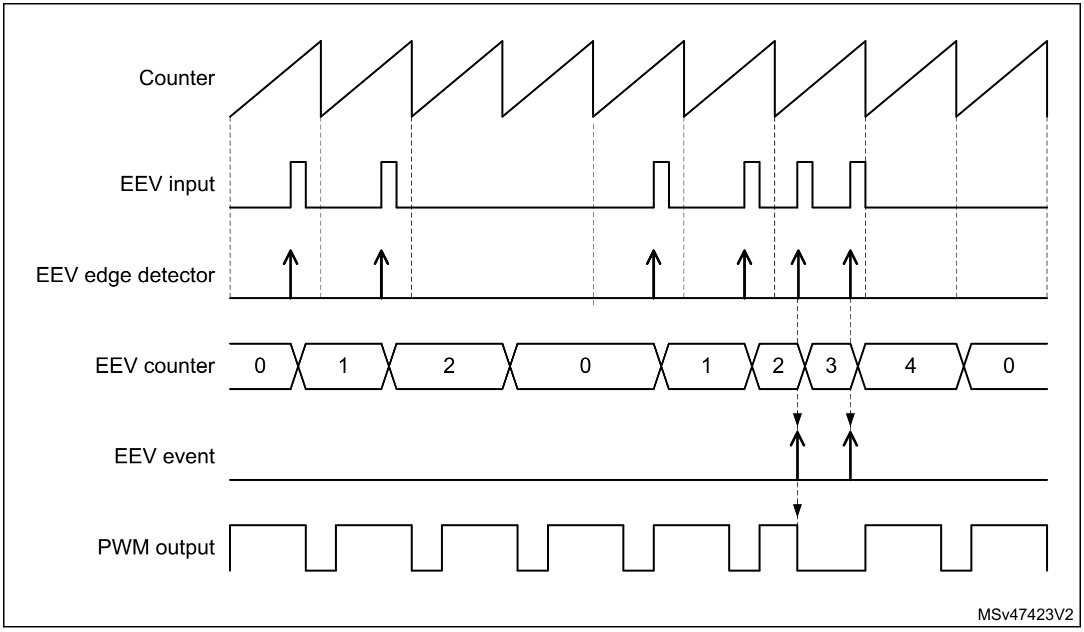
- when the EEVARSTM bit is reset, the external event counter is reset on each reset/roll-over event: the external event is active only if it appears several times within a given PWM period
- when the EEVARSTM/ bit is set, the external event counter is reset only if the event did not on appear during the last PWM period. This a cumulative mode, where the event must occur at least once during multiple PWM period, as shown on *Figure 227* below.

The external event counter must be enabled after having programmed the counter value (the EEVACE bit must be set after having written the EEVACNT[5:0] bits).

Once the counter is enabled, the EEVACNT[5:0] bits can then be changed on-the-fly at any time. The new value is taken into account on the following reset/rollover event as per the EEVARSTM bit programming, or after a software reset (EEVACRES bit set).

The EEVASEL[3:0]bits must not be modified once the EEVACE bit is set.

Figure 227. External event counter cumulative mode (EEVxRSTM = 1, EEVxCNT = 2)



27.3.10 Delayed protection

The HRTIM features specific protection schemes, typically for resonant converters when it is necessary to shut down the PWM outputs in a delayed manner, either once the active pulse is completed or once a push-pull period is completed. These features are enabled with DLYPRTEN bit in the HRTIM_OUTxR register, and are using specific external event channels.

Delayed idle

In this mode, the active pulse is completed before the protection is activated. The selected external event causes the output to enter in idle mode at the end of the active pulse (defined by an output reset event in HRTIM_RSTx1R or HRTIM_RSTx2R).

Once the protection is triggered, the idle mode is permanently maintained but the counter continues to run, until the output is re-enabled. Tx1OEN and Tx2OEN bits are not affected by the delayed idle entry. To exit from delayed idle and resume operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1. The output state changes on the first transition to an active state following the output enable command.

Note: The delayed idle mode cannot be exited immediately after having been entered, before the active pulse is completed: it is mandatory to make sure that the outputs are in idle state before resuming the run mode. This can be done by waiting up to the next period, for instance, or by polling the O1CPY and/or O2CPY status bits in the TIMxISR register.

The delayed idle mode can be applied to a single output (DLYPRT[2:0] = x00 or x01) or to both outputs (DLYPRT[2:0] = x10).

An interrupt or a DMA request can be generated in response to a Delayed Idle mode entry. The DLYPRT flag in HRTIM_TIMxISR is set as soon as the external event arrives, independently from the end of the active pulse on output.

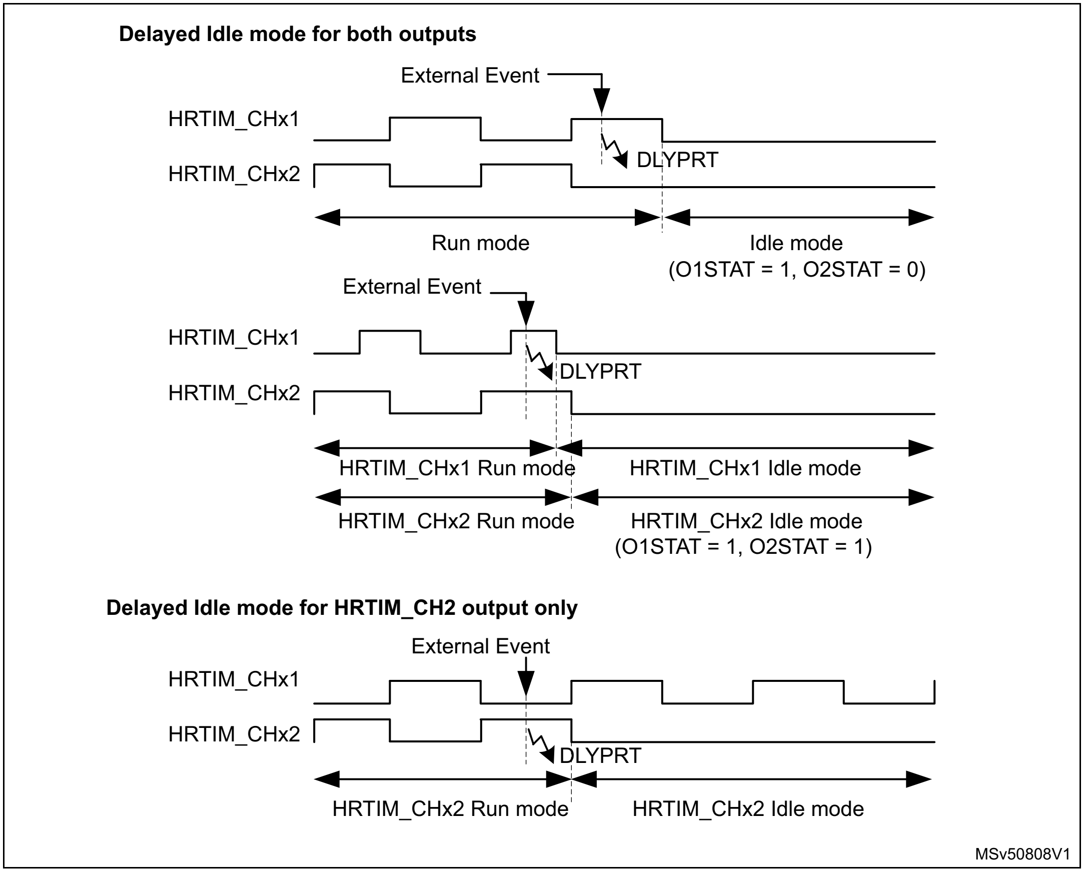
When the Delayed Idle mode is triggered, the output states can be determined using O1STAT and O2STAT in HRTIM_TIMxISR. Both status bits are updated even if the delayed idle is applied to a single output. When the push-pull mode is enabled, the IPPSTAT flag in HRTIM_TIMxISR indicates during which period the delayed protection request occurred.

This mode is available whatever the timer operating mode (regular, push-pull, deadtime). It is available with 2 external events only:

- hrtim_eev6 and hrtim_eev7 for timer A, B and C
- hrtim_eev8 and hrtim_eev9 for timer D, E and F

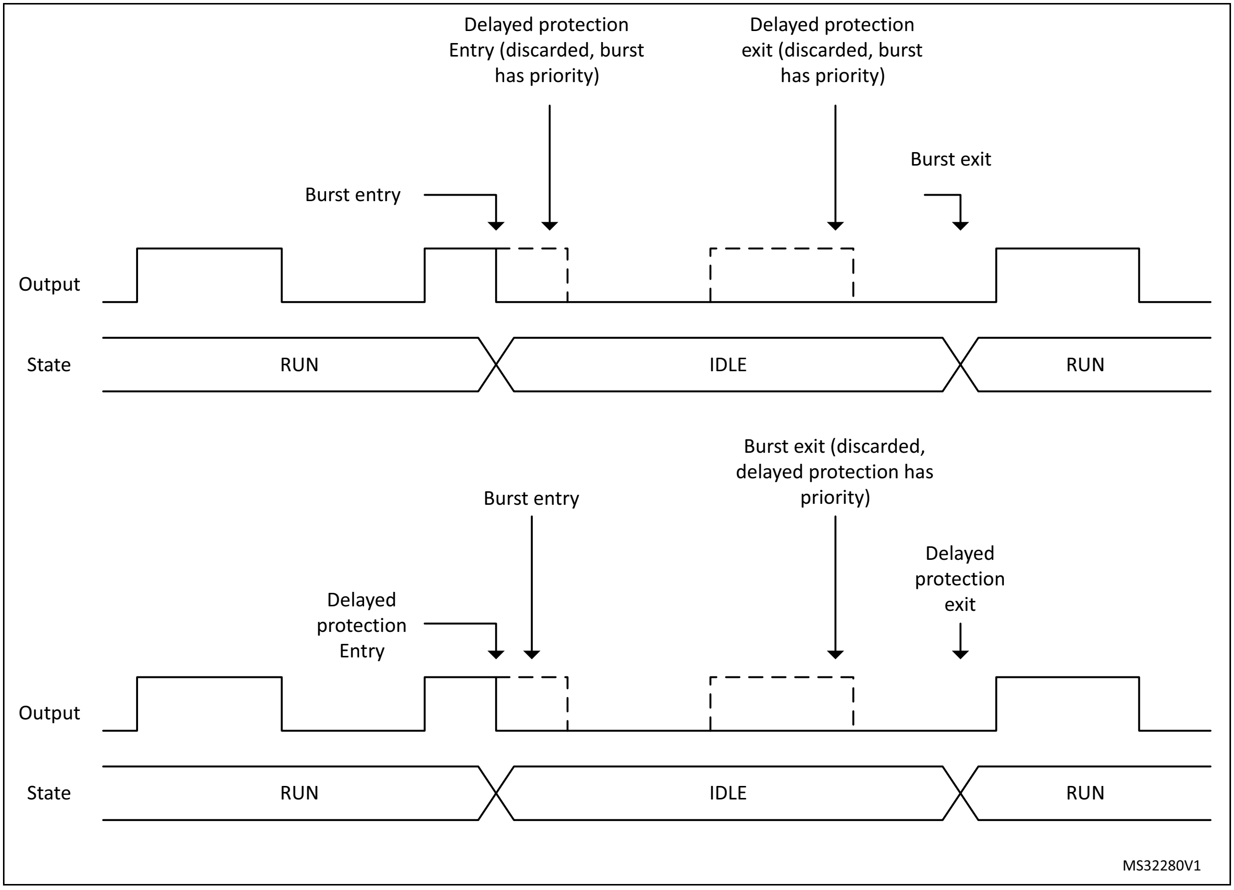
The delayed protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxEN bit is reset, until the TxyOEN bits are set.

Figure 228. Delayed Idle mode entry



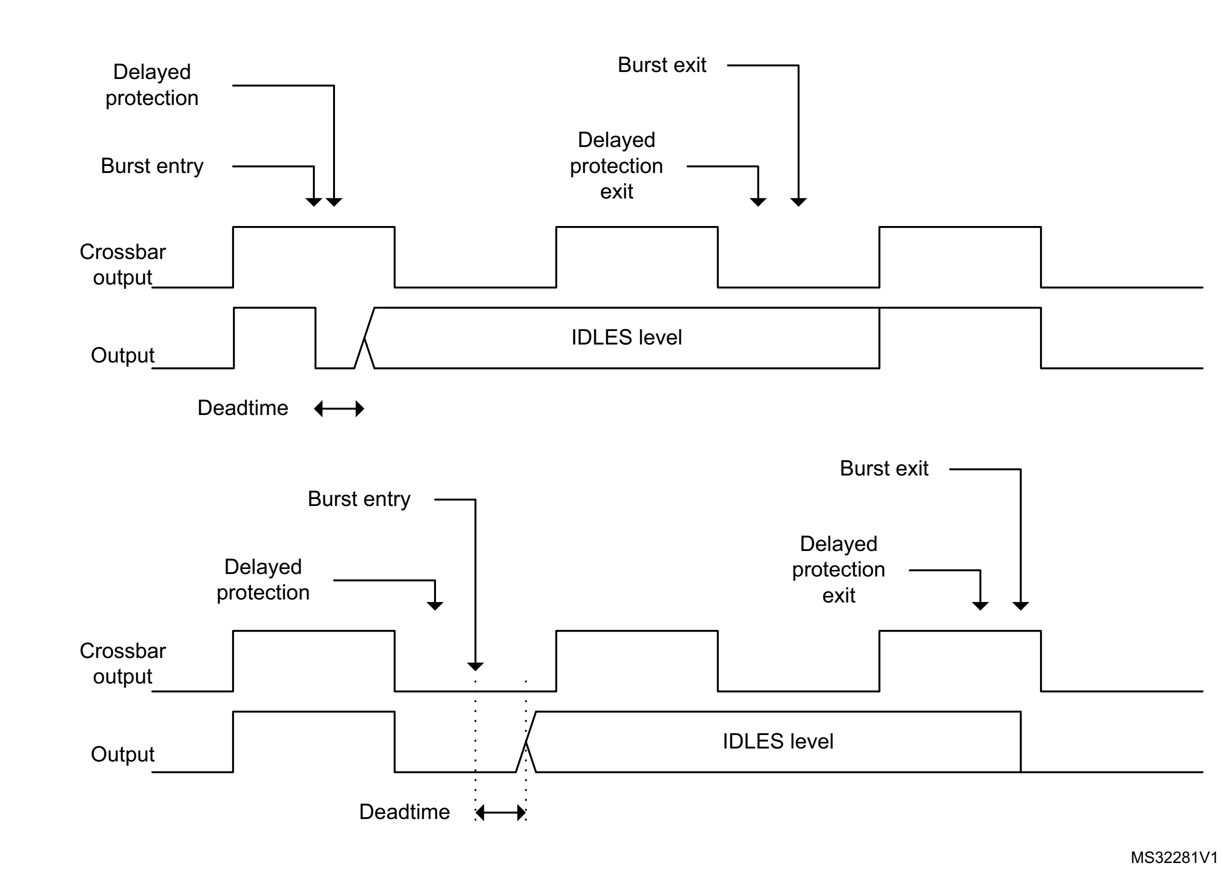
The delayed idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the delayed idle protection has been triggered. On the contrary, If the delayed protection is exited while the burst mode is active, the burst mode is resumed normally and the output is maintained in the idle state until the burst mode exits. *Figure 229* gives an overview of these different scenarios.

Figure 229. Burst mode and delayed protection priorities (DIDL = 0)


Delayed protection Entry (discarded, burst has priority)
Delayed protection exit (discarded, burst has priority)
Burst exit
Burst entry
Output
State
RUN
IDLE
RUN
Burst exit (discarded, delayed protection has priority)
Burst entry
Delayed protection exit
Delayed protection Entry
Output
State
RUN
IDLE
RUN
MS32280V1

The same priorities are applied when the delayed burst mode entry is enabled (DIDL bit set), as shown on *Figure 230* below.

Figure 230. Burst mode and delayed protection priorities (DIDL = 1)



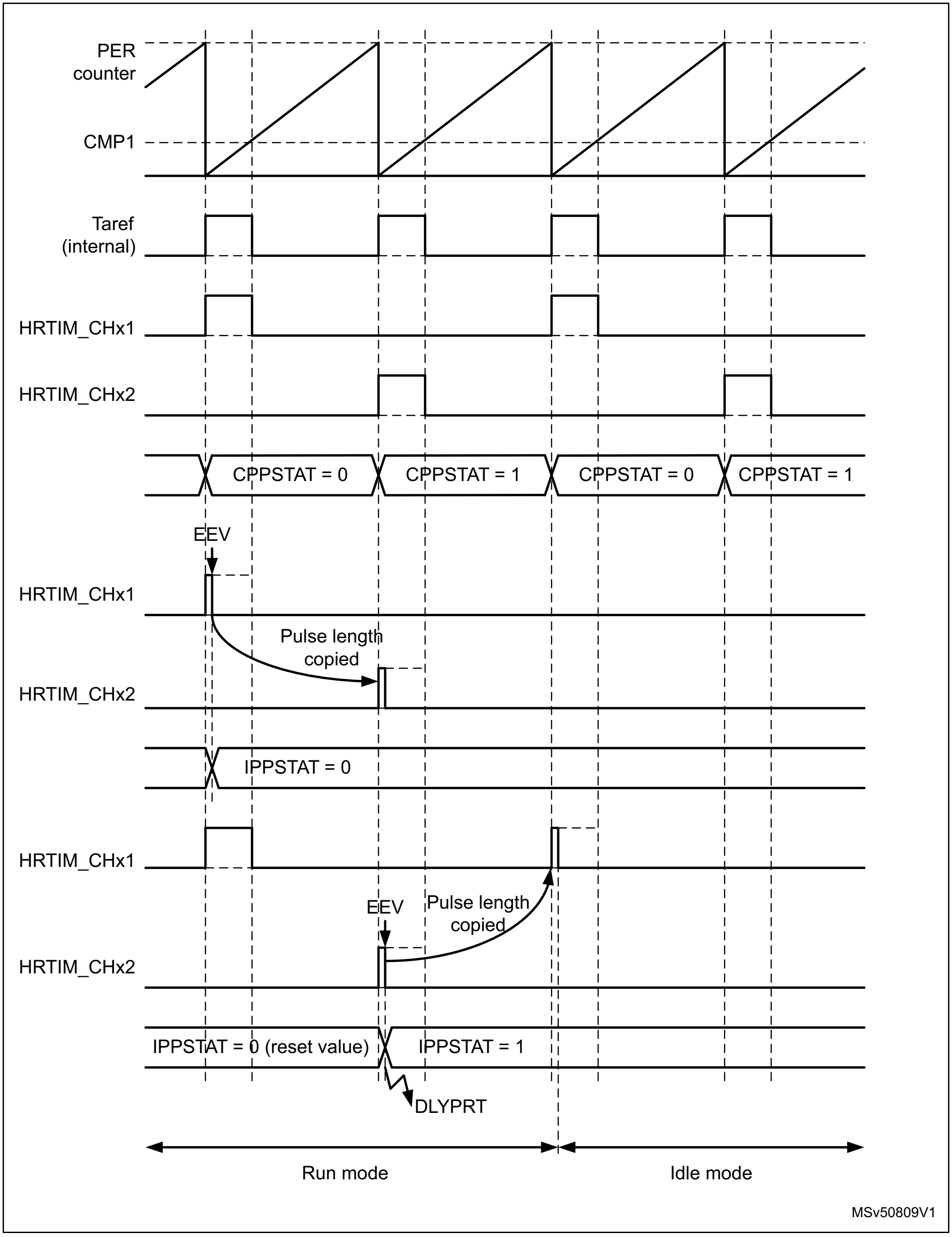
Balanced idle

Only available in push-pull mode, the balanced idle allows to have a balanced pulsewidth on the two outputs when one of the active pulse is shortened due to a protection. The pulsewidth, terminated earlier than programmed, is copied on the alternate output, then the two outputs are put in idle state, until the normal operation is resumed by software. This mode is enabled by writing x11 in DLYPRT[2:0] bitfield in HRTIM_OUTxR.

This mode is available with only 2 external events:

- hrtim_eev6 and hrtim_eev7 for timer A, B and C
- hrtim_eev8 and hrtim_eev9 for timer D, E and F

Figure 231. Balanced Idle protection example



When the balanced Idle mode is enabled, the selected external event triggers a capture of the counter value into the compare 4 active register (this value is not user-accessible). The push-pull is maintained for one additional period so that the shorten pulse can be repeated: a new output reset event is generated while the regular output set event is maintained.

The Idle mode is then entered and the output takes the level defined by IDLESx bits in the HRTIM_OUTxR register. The balanced idle mode entry is indicated by the DLYPRT flag, while the IPPSTAT flag indicates during which period the external event occurred, to determine the sequence of shorten pulses (A1 then A2 or vice versa).

The timer operation is not interrupted (the counter continues to run).

To enable the balanced idle mode, it is necessary to have the following initialization:

- – timer operating in continuous mode (CONT = 1)
- – Push-pull mode enabled
- – HRTIM_CMP4xR must be set to 0 and the content transferred into the active register (for instance by forcing a software update)
- – DELCMP4[1:0] bit field must be set to 00 (auto-delayed mode disabled)
- – DLYPRT[2:0] = x11 (delayed protection enable)

Note: The HRTIM_CMP4xR register must not be written during a balanced idle operation. The CMP4 event is reserved and cannot be used for another purpose.

In balanced idle mode, it is recommended to avoid multiple external events or software-based reset events causing an output reset. If such an event arrives before a balanced idle request within the same period, it causes the output pulses to be unbalanced (1st pulse length defined by the external event or software reset, while the 2nd pulse is defined by the balanced idle mode entry).

The minimum pulsewidth that can be handled in balanced idle mode is 4 f_{HRTIM} clock periods (0x80 when CKPSC[2:0] = 0, 0x40 if CKPSC[2:0] = 1, 0x20 if CKPSC[2:0] = 2,...).

If the capture occurs before the counter has reached this minimum value, the current pulse is extended up to 4 f_{HRTIM} clock periods before being copied into the secondary output. In any case, the pulsewidths are always balanced.

Tx1OEN and Tx2OEN bits are not affected by the balanced idle entry. To exit from balanced idle and resume the operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1 simultaneously. The output state changes on the first active transition following the output enable.

It is possible to resume operation similarly to the delayed idle entry. For instance, if the external event arrives while output 1 is active (delayed idle effective after output 2 pulse), the re-start sequence can be initiated for output 1 first. To do so, it is necessary to poll CPPSTAT bit in the HRTIM_TIMxISR register. Using the above example (IPPSTAT flag equal to 0), the operation is resumed when CPPSTAT bit is 0.

In order to have a specific re-start sequence, it is possible to poll the CPPSTAT to know which output is active first. This allows, for instance, to re-start with the same sequence as the idle entry sequence: if the external event arrives during output 1 active, the re-start sequence is initiated when the output 1 is active (CPPSTAT = 0).

Note: The balanced idle mode must not be disabled while a pulse balancing sequence is on-going. It is necessary to wait until the CMP4 flag is set, thus indicating that the sequence is completed, to reset the DLYPRTEN bit.

The balanced idle protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxCEN bit is reset, until TxyOEN bits are set.

Balanced idle can be used together with the burst mode under the following conditions:

- TxBM bit must be reset (counter clock maintained during the burst, see *Section 27.3.15*),
- No balanced idle protection must be triggered while the outputs are in a burst idle state.

The balanced idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the balanced idle protection has been triggered. On the contrary, if the delayed protection is exited while the burst mode is active, the burst mode is resumed normally.

Note: *Although the output state is frozen in idle mode, a number of events are still generated on the auxiliary outputs (see Section 27.3.18) during the idle period following the delayed protection:*
- Output set/reset interrupt or DMA requests
- External event filtering based on output signal
- Capture events triggered by set/reset

Balanced idle automatic resuming

The balanced Idle mode can be configured to have an automatic resuming of operation after a trigger.

Once the shorten pulse has been copied to the alternate output, the pulse width is reset to its original value and the timer resumes operation: the two outputs keep on being in RUN mode.

This is enabled by setting the BIAR bit in the HRTIM_OUTxR register.

This mode must be used only when the period in HRTIM_PERxR is greater than 6 periods of the fHRTIM clock, that is 0xC0 if CKPSC[2:0] = 0, 0x60 if CKPSC[2:0] = 1, 0x30 if CKPSC[2:0] = 2, ...

Note: *This bit is only significant if DLYPRT[2:0] = 011 or 111, it is ignored otherwise.*

Note: *In balanced idle automatic resuming mode, it is mandatory to set the IDLES state to inactive.*

27.3.11 Register preload and update management

Most of HRTIM registers are buffered and can be preloaded if needed. Typically, this allows to prevent the waveforms from being altered by a register update not synchronized with the active events (set/reset).

When the preload mode is enabled, accessed registers are shadow registers. Their content is transferred into the active register after an update request, either software or synchronized with an event.

By default, PREEN bits in HRTIM_MCR and HRTIM_TIMxCR registers are reset and the registers are not preloaded: any write directly updates the active registers. If PREEN bit is reset while the timer is running and preload was enabled, the content of the preload registers is directly transferred into the active registers.

Each timing unit and the master timer have their own PREEN bit. If PRREN is set, the preload registers are enabled and transferred to the active register only upon an update event.

There are two options to initialize the timer when the preload feature is needed:

- Enable PREEN bit at the very end of the timer initialization to have the preload registers transferred into the active registers before the timer is enabled (by setting MCEN and TxCEN bits).
- enable PREEN bit at any time during the initialization and force a software update immediately before starting.

Table 226 lists the registers which can be preloaded, together with a summary of available update events.

Table 226. HRTIM preloadable control registers and associated update sources

Timer	Preloadable registers	Preload enable	Update sources
Master timer	HRTIM_DIER HRTIM_MPER HRTIM_MREP HRTIM_MCMP1R HRTIM_MCMP2R HRTIM_MCMP3R HRTIM_MCMP4R	PREEN bit in HRTIM_MCR	Software Repetition event Burst DMA event Repetition event following a burst DMA event
Timer x x = A..F	HRTIM_TIMxDIER HRTIM_TIMxPER HRTIM_TIMxREP HRTIM_TIMxCMP1R HRTIM_TIMxCMP1CR HRTIM_TIMxCMP2R HRTIM_TIMxCMP3R HRTIM_TIMxCMP4R HRTIM_DTxR HRTIM_SETx1R HRTIM_RSTx1R HRTIM_SETx2R HRTIM_RSTx2R HRTIM_RSTxR	PREEN bit in HRTIM_TIMxCR	Software TIMx repetition event TIMx reset event Burst DMA event Update event from other timers (TIMy, master) Update event following a burst DMA event Update enable inputs hrtim_upd_en[3:1] Update event following an update enable input following an update event on hrtim_upd_en[3:1] inputs
HRTIM Common	HRTIM_ADC1R HRTIM_ADC2R HRTIM_ADC3R HRTIM_ADC4R	TIMx or master timer Update, depending on ADxUSRC[2:0] bits in HRTIM_CR1, if PREEN = 1 in the selected timer	

The master timer has 4 update options:

1. Software: writing 1 into MSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is cancelled.
2. Update done when the master counter rolls over and the master repetition counter is equal to 0. This is enabled when MREPU bit is set in HRTIM_MCR.
3. Update done once burst DMA is completed (see *Section 27.3.23* for details). This is enabled when BRSTDMA[1:0] = 01 in HRTIM_MCR. It is possible to have both MREPU=1 and BRSTDMA=01.
   Note: The update can take place immediately after the end of the burst sequence if SWU bit is set (i.e. forced update mode). If SWU bit is reset, the update is done on the next update event following the end of the burst sequence.
4. Update done when the master counter rolls over following a burst DMA completion. This is enabled when BRSTDMA[1:0] = 10 in HRTIM_MCR.

An interrupt or a DMA request can be generated by the master update event.

Each timer (TIMA..F) can also have the update done as follows:

- By software: writing 1 into TxSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is canceled.
- Update done when the counter rolls over and the repetition counter is equal to 0. This is enabled when TxREPU bit is set in HRTIM_TIMxCR.
- Update done when the counter is reset or rolls over in continuous mode. This is enabled when TxRSTU bit is set in HRTIM_TIMxCR. This is used for a timer operating in single-shot mode, for instance.
- Update done once a burst DMA is completed. This is enabled when UPDGAT[3:0] = 0001 in HRTIM_TIMxCR.
- Update done on the update event following a burst DMA completion (the event can be enabled with TxRSTU, TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0010 in HRTIM_TIMxCR.
- Update done when receiving a request on hrtim_upd_en[3:1]. This is enabled when UPDGAT[3:0] = 0011, 0100, 0101 in HRTIM_TIMxCR.
- Update done on the update event following a request on hrtim_upd_en[3:1] (the event can be enabled with TxRSTU, TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0110, 0111, 1000 in HRTIM_TIMxCR
- Update done synchronously with any other timer or master update (for instance TIMA can be updated simultaneously with TIMB). This is used for converters requiring several timers, and is enabled by setting bits MSTU and TxU in HRTIM_TIMxCR register.

The update enable inputs hrtim_upd_en[3:1] allow to have an update event synchronized with on-chip events coming from the general-purpose timers. These inputs are rising-edge sensitive.

Table 209 lists the connections between update enable inputs and the on-chip sources.

This allows to synchronize low frequency update requests with high-frequency signals (for instance an update on the counter roll-over of a 100 kHz PWM that has to be done at a 100 Hz rate).

Note: The update events are synchronized to the prescaler clock when CKPSC[2:0] > 5.

The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) can either be taken into account immediately or re-synchronized with the timers reset/roll-over event. This is done with the RSYNCU bit in the HRTIM_TIMxCR register, as show on *Figure 232* below):

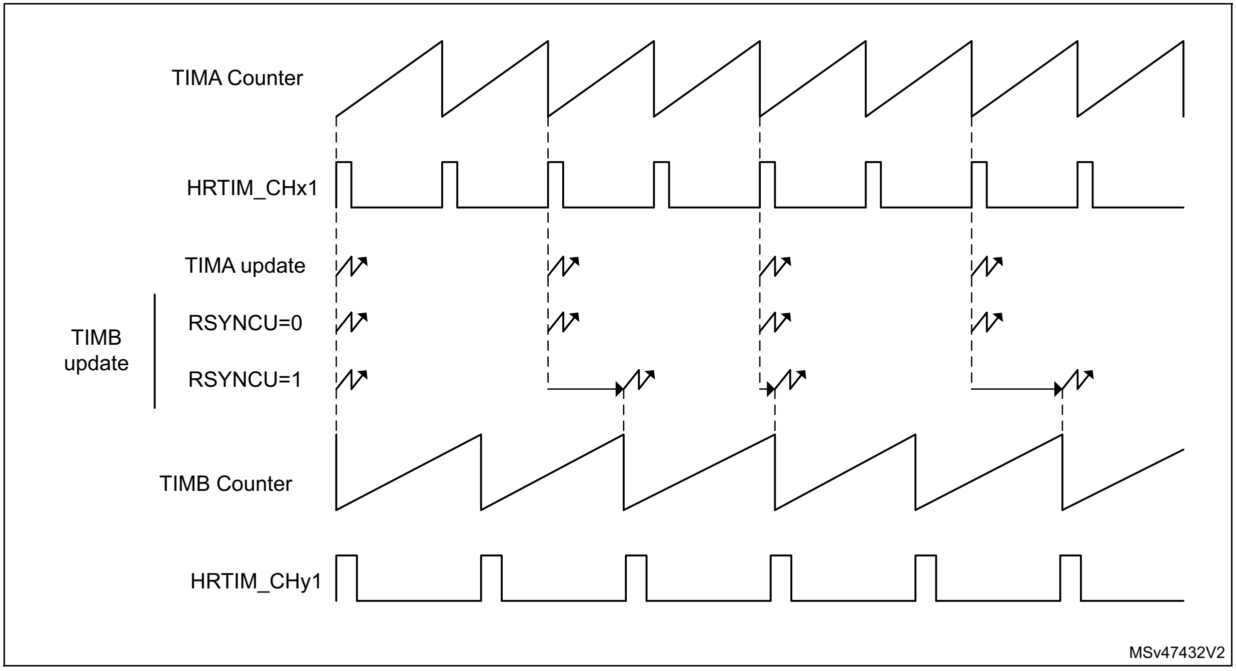
– RSYNCU = 0: The update coming from adjacent timers is taken into account immediately

– RSYNCU = 1: The update coming from adjacent timers is taken into account on the following reset/roll-over event.

The RSYNCU bit is significant only when UPDGAT[3:0] = 0000, it is ignored otherwise.

An interrupt or a DMA request can be generated by the Timx update event.

Figure 232. Resynchronized timer update (TAU=1 in HRTIM_TIMBCR)



MUDIS and TxUDIS bits in the HRTIM_CR1 register allow to temporarily disable the transfer from preload to active registers, whatever the selected update event. This allows to modify several registers in multiple timers. The regular update event takes place once these bits are reset.

MUDIS and TxUDIS bits are all grouped in the same register. This allows the update of multiple timers (not necessarily synchronized) to be disabled and resumed simultaneously.

The following example is a practical use case. A first power converter is controlled with the master, TIMB and TIMC. TIMB and TIMC must be updated simultaneously with the master timer repetition event. A second converter works in parallel with TIMA, TIMD and TIME, and TIMD, TIME must be updated with TIMA repetition event.

First converter

In HRTIM_MCR, MREPU bit is set: the update occurs at the end of the master timer counter repetition period. In HRTIM_TIMBCR and HRTIM_TIMCCR, MSTU bits are set to have TIMB and TIMC timers updated simultaneously with the master timer.

When the power converter set-point has to be adjusted by software, MUDIS, TBUDIS and TCUDIS bits of the HRTIM_CR register must be set prior to write accessing registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, MUDIS, TBUDIS and TCUDIS bits must be reset. The transfer from preload to active registers is done as soon as the master repetition event occurs.

Second converter

In HRTIM_TIMACR, TAREPU bit is set: the update occurs at the end of the timer A counter repetition period. In HRTIM_TIMDCR and HRTIM_TIMECR, TAU bits are set to have TIMD and TIME timers updated simultaneously with timer A.

When the power converter set-point has to be adjusted by software, TAUDIS, TDUDIS and TEUDIS bits of the HRTIM_CR register must be set prior to write accessing the registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, TAUDIS, TDUDIS and TEUDIS bits can be reset: the transfer from preload to active registers is done as soon as the timer A repetition event occurs.

27.3.12 PWM mode with “greater than” comparison

A specific no-latency update mode is available for PWM signals generated with the CMP1 and CMP3 registers. It allows to have a new duty cycle value applied as soon as possible within the PWM cycle, without having to wait the completion of the current PWM period. This reduces the overall delay time in software control loops. As shown on *Figure 233* below, this eventually allows to have:

- an early turn-off of the output if the new compare value is below the current counter value and the current compare value is above the counter, at the time the new value is written.
- an early turn-on of the output, re-enabling the output if the new compare value is above the counter value and the current compare value is above the counter, at the time the new value is written.

The output signal is left unchanged when the new compare value and current compare value are both below the counter.

This feature is only available for CMP1 or CMP3 RESET events, and is enabled using the GTCMP1 and GTCMP3 enable bits in the HRTIM_TIMxCR2 register.

The preload mechanism is inactive for a compare register when the corresponding GTCMPx bit is set, whatever the PREEN bit value. This mode is intended to have the new compare value taken into account as soon as possible after a new value write, without waiting for the preload to active register transfer.

These bits are defining the compare 1 and compare 3 operating modes as following

- GTCMPx = 0: the compare x event is generated when the counter is equal to the compare value (compare match mode). If the compare value is changed on-the-fly, the compare event may not be generated.
- GTCMPx = 1: the compare x event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

The “greater than” compare mode causes the crossbar to act differently depending on comparison result. Let’s consider the CMP1 event is doing an output RESET. When the new compare value is written, two cases are considered

- If the new compare value is below the counter value, the RESET event is issued and can eventually cause an early turn-OFF
- If the new compare value is above the counter value, a SET event is generated so as to re-arm the output value before it is actually RESET when the counter exceeds the counter value (early turn-ON).

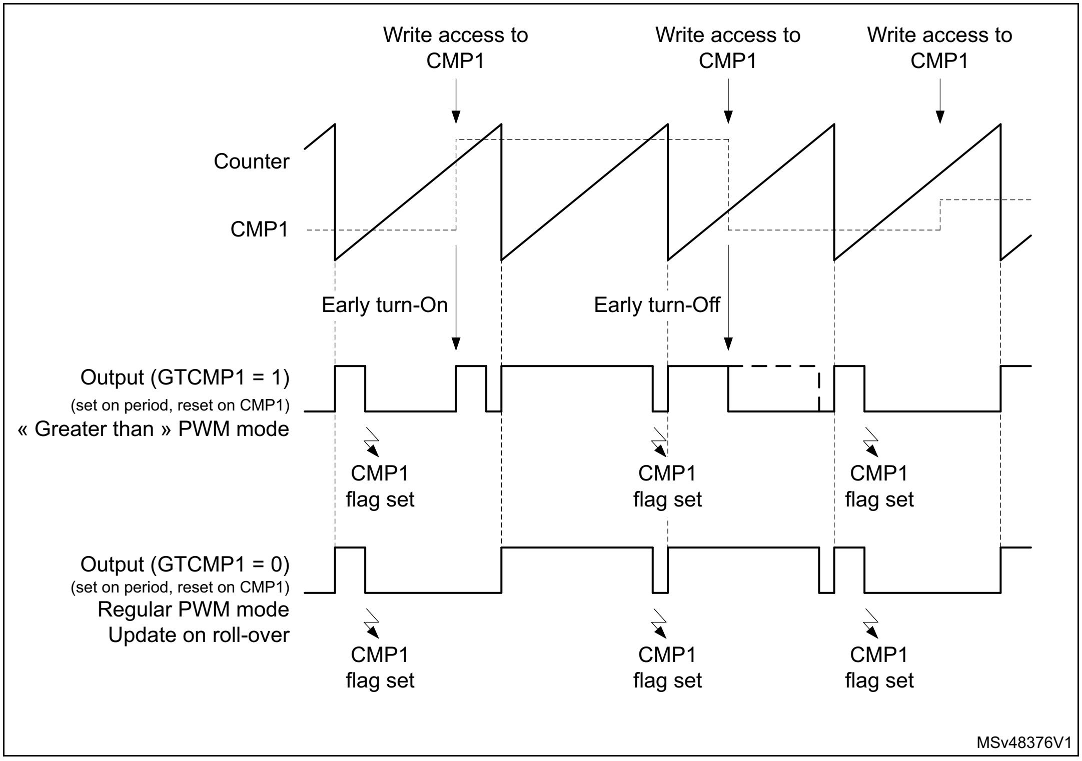
The “greater than” compare mode is supported for both SET and RESET actions.

The “greater than” compare mode must only be used for the following configuration:

1. In the fixed frequency configuration, the period event must trigger the output set and the “greater than” compare triggers the output reset (or vice versa the period must trigger the reset if the “greater-than” compare triggers the set).
2. For variable frequency configuration, the event selected as counter reset source must also be selected as set or reset source for the timer output (opposite direction as the “greater than” compare event).

Note: The “greater-than” modes must not be used when the CMP1 and/or CMP3 modes are controlled by hardware in half and interleaved modes.

Figure 233. Early turn-ON and early turn-OFF behavior in “greater than” PWM mode



The immediate update mode implies that the content of the preload register is transferred into the active register at the very same time the register is written. When GTCMP1 and/or GTCMP3 bits are set, their respective preload mechanism is disabled (for HRTIM_TIMxCMP1 and/or HRTIM_TIMxCMP3 registers), whatever the PREEN bit value.

Note: The compare interrupt flags (CMP1 and CMP3 in HRTIM_TIMxISR) are not generated in case of late turn-ON and early turn-OFF, as shown on Figure 233.

Note: The “Greater than” comparison must not be done on both CMP1 and CMP3 for the same output (GTCMP1 and GTCMP3 bits must not be set simultaneously).

27.3.13 Events propagation within or across multiple timers

The HRTIM offers many possibilities for cascading events or sharing them across multiple timing units, including the master timer, to get full benefits from its modular architecture. These are key features for converters requiring multiple synchronized outputs.

This section summarizes the various options and specifies whether and how an event is propagated within the HRTIM.

TIMx update triggered by the master timer update

The sources listed in *Table 227* are generating a master timer update. The table indicates if the source event can be used to trigger a simultaneous update in any of TIMx timing units.

Operating condition: MSTU bit is set in HRTIM_TIMxCR register.

Table 227. Master timer update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	BRSTDMA[1:0] = 01	No	Must be done in TIMxCR (UPDGAT[3:0] = 0001)
Roll-over event following a burst DMA end	BRSTDMA[1:0] = 10	Yes	-
Repetition event caused by a counter roll-over	MREPU = 1	Yes	-
Repetition event caused by a counter reset (from HRTIM_SCIN or software)		No	-
Software update	MSWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx update triggered by the TIMy update

The sources listed in *Table 228* are generating a TIMy update. The table indicates if the given event can be used to trigger a simultaneous update in another or multiple TIMx timers.

Operating condition: TyU bit set in HRTIM_TIMxCR register (source = TIMy and destination = TIMx).

Table 228. TIMx update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0001)
Update caused by the update enable inputs hrtim_upd_en[3:1]	UPDGAT[3:0] = 0011, 0100, 0101	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0011, 0100, 0101
Master update	MSTU = 1 in HRTIM_TIMyCR	No	Must be done with MSTU = 1 in HRTIM_TIMxCR

Another TIMx update
(TIMz>TIMy>TIMx)
	TzU=1 in HRTIM_TIMyCR
	TyU=1 in TIMxCR
		No	Must be done with TzU=1 in HRTIM_TIMxCR
			TzU=1 in HRTIM_TIMyCR

Repetition event caused by a counter roll-over	TyREPU = 1	Yes	-
Repetition event caused by a counter reset	TyREPU = 1	-	Refer to counter reset cases below
Counter roll-over	TyRSTU = 1	Yes	-

Table 228. TIMx update event propagation (continued)

Source	Condition	Propagation	Comment
Counter software reset	TyRST=1 in HRTIM_CR2	No	Can be done simultaneously with update in HRTIM_CR2 register
Counter reset caused by a TIMz compare	TIMzCMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by external events	EXTEVNTn in HRTIM_RSTyR	Yes	-
Counter reset caused by a master compare or a master period	MSTCMPn or MSTPER in HRTIM_RSTyR	Yes	-
Counter reset caused by a TIMy compare	CMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by an update	UPDT in HRTIM_RSTyR	No	Propagation would result in a lock-up situation (update causing reset causing update)
Counter reset caused by HRTIM_SCIN	SYNCRSTy in HRTIM_TIMyCR	No	-
Software update	TySWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx counter reset causing a TIMx update

Table 229 lists the counter reset sources and indicates whether they can be used to generate an update.

Operating condition: TxRSTU bit in HRTIM_TIMxCR register.

Table 229. Reset events able to generate an update

Source	Condition	Propagation	Comment
Counter roll-over	-	Yes	-
Update event	UPDT in HRTIM_RSTxR	No	Propagation would result in a lock-up situation (update causing a reset causing an update)
External event	EXTEVNTn in HRTIM_RSTxR	Yes	-
TIMy compare	TIMyCMPn in HRTIM_RSTxR	Yes	-
Master compare	MSTCMPn in HRTIM_RSTxR	Yes	-
Master period	MSTPER in HRTIM_RSTxR	Yes	-
Compare 2 and 4	CMPn in HRTIM_RSTxR	Yes	-
Software	TxRST=1 in HRTIM_CR2	Yes	-
HRTIM_SCIN	SYNCRSTx in HRTIM_TIMxCR	Yes	-

TIMx update causing a TIMx counter reset

Table 230 lists the update event sources and indicates whether they can be used to generate a counter reset.

Operating condition: UPDT bit set in HRTIM_RSTxR.

Table 230. Update event propagation for a timer reset

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	Yes	-
Update caused by the update enable inputs hrtim_upd_en[3:1]	UPDGAT[3:0] = 0011, 0100, 0101	Yes	-
Master update caused by a roll-over after a burst DMA	MSTU = 1 in HRTIM_TIMxCR BRSTDMA[1:0] = 10 in HRTIM_MCR	Yes	-

Table 230. Update event propagation for a timer reset (continued)

Source	Condition	Propagation	Comment

Master update caused by a repetition event following a roll-over	MSTU = 1 in HRTIM_TIMxCR
	MREPU = 1 in HRTIM_MCR
		Yes	-

Master update caused by a repetition event following a counter reset (software or due to HRTIM_SCIN)		No	-

Software triggered master timer update	MSTU = 1 in HRTIM_TIMxCR
	MSWU = 1
	in HRTIM_CR2
		No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMy update caused by a TIMy counter roll-over	TyU = 1 in HRTIM_TIMxCR
	TyRSTU = 1 in HRTIM_TIMyCR
		Yes	-

TIMy update caused by a TIMy repetition event	TyU = 1 in HRTIM_TIMxCR
	TyREPU = 1 in HRTIM_TIMyCR
		Yes	-

TIMy update caused by an external event or a TIMy compare (through a TIMy reset)	TyU = 1 in HRTIM_TIMxCR
	TyRSTU = 1 in HRTIM_TIMyCR
	EXTEVNTn or CMP4/2
	in HRTIM_RSTyCR
		Yes	-

TIMy update caused by sources other than those listed above	TyU = 1 in HRTIM_TIMxCR	No	-
Repetition event following a roll-over	TxREPU = 1 in HRTIM_TIMxCR	Yes	-
Repetition event following a counter reset		No	-
Timer reset	TxRSTU = 1 in HRTIM_TIMxCR	No	Propagation would result in a lock-up situation (reset causing an update causing a reset)
Software	TxSWU in HRTIM_CR2	No	-

27.3.14 Output management

Each timing unit controls a pair of outputs. The outputs have three operating states:

- RUN: this is the main operating mode, where the output can take the active or inactive level as programmed in the crossbar unit.
- IDLE: this state is the default operating state after an HRTIM reset, when the outputs are disabled by software or during a burst mode operation (where outputs are temporary disabled during a normal operating mode; refer to *Section 27.3.15* for more details). It is either permanently active or inactive.
- FAULT: this is the safety state, entered in case of a shut-down request on FAULTx inputs. It can be permanently active, inactive or Hi-Z.

The output status is indicated by TxyOEN bit in HRTIM_OENR register and TxyODS bit in HRTIM_ODSR register, as in *Table 231*.

Table 231. Output state programming, x= A..F, y = 1 or 2

TxyOEN (control/status) (set by software, cleared by hardware)	TxyODS (status)	Output operating state
1	x	RUN
0	0	IDLE
0	1	FAULT

TxyOEN bit is both a control and a status bit: it must be set by software to have the output in RUN mode. It is cleared by hardware when the output goes back in IDLE or FAULT mode. When TxyOEN bit is cleared, TxyODS bit indicates whether the output is in the IDLE or FAULT state. A third bit in the HRTIM_ODISR register allows to disable the output by software.

Figure 234. Output management overview

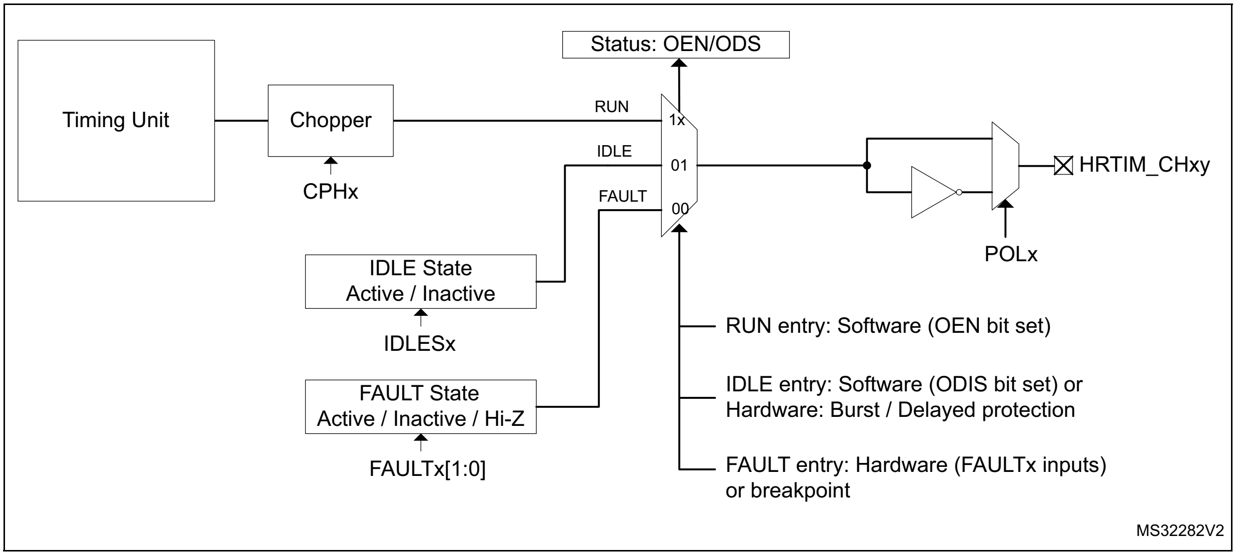
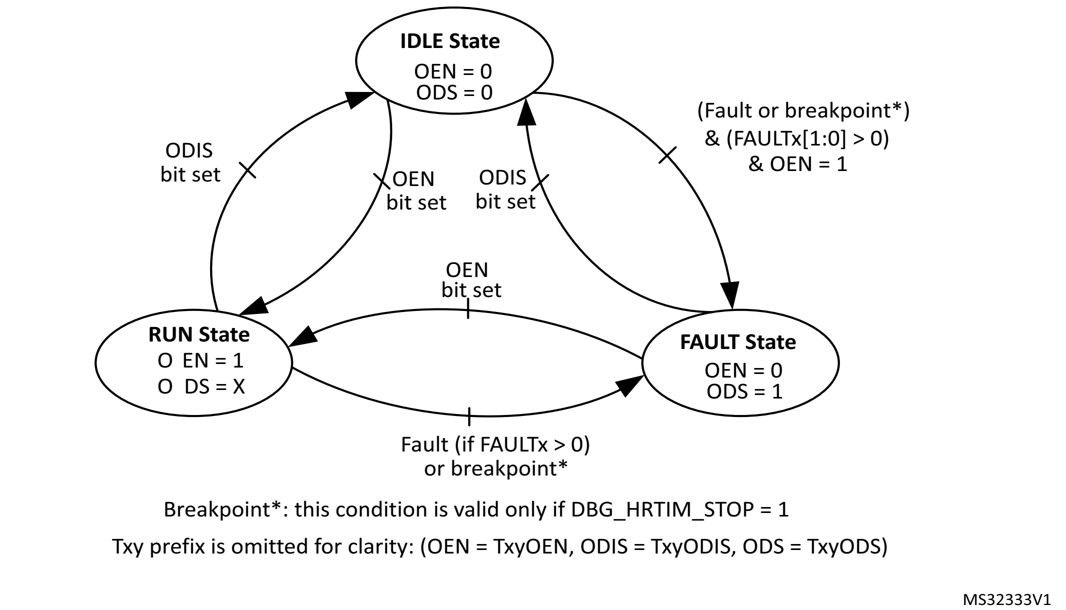


Figure 235 summarizes the bit values for the three states and how the transitions are triggered. Faults can be triggered by any external or internal fault source, as listed in

Section 27.3.17, while the Idle state can be entered when the burst mode or delayed protections are active.

Figure 235. HRTIM output states and transitions



Breakpoint*: this condition is valid only if DBG_HRTIM_STOP = 1

Txy prefix is omitted for clarity: (OEN = TxyOEN, ODIS = TxyODIS, ODS = TxyODS)

The FAULT and IDLE levels are defined as active or inactive. Active (or inactive) refers to the level on the timer output that causes a power switch to be closed (or opened for an inactive state).

The IDLE state has the highest priority: the transition FAULT → IDLE is possible even if the FAULT condition is still valid, triggered by ODIS bit set.

The FAULT state has priority over the RUN state: if TxyOEN bit is set simultaneously with a fault event, the FAULT state is entered. The condition is given on the transition IDLE → FAULT, as in *Figure 235*: fault protection needs to be enabled (FAULTx[1:0] bits = 01, 10, 11) and the Txy OEN bit set with a fault active (or during a breakpoint if DBG_HRTIM_STOP = 1).

The output polarity is programmed using POLx bits in HRTIM_OUTxR. When POLx = 0, the polarity is positive (output active high), while it is active low in case of a negative polarity (POLx = 1). Practically, the polarity is defined depending on the power switch to be driven (PMOS versus NMOS) or on a gate driver polarity.

The output level in the FAULT state is configured using FAULTx[1:0] bits in HRTIM_OUTxR, for each output, as follows:

- 00: output never enters the fault state and stays in RUN or IDLE state
- 01: output at active level when in FAULT
- 10: output at inactive level when in FAULT
- 11: output is tri-stated when in FAULT. The safe state must be forced externally with pull-up or pull-down resistors, for instance.

Note: FAULTx[1:0] bits must not be changed as long as the outputs are in FAULT state.

The level of the output in IDLE state is configured using IDLESx bit in HRTIM_OUTxR, as follows:

- 0: output at inactive level when in IDLE
- 1: output at active level when in IDLE

When TxyOEN bit is set to enter the RUN state, the output is immediately connected to the crossbar output. If the timer clock is stopped, the level is either inactive (after an HRTIM reset) or corresponds to the RUN level (when the timer is stopped and the output disabled).

During the HRTIM initialization, the output level can be prepositioned prior to have it in RUN mode, using the software forced output set and reset in the HRTIM_SETx1R and HRTIM_RSTx1R registers.

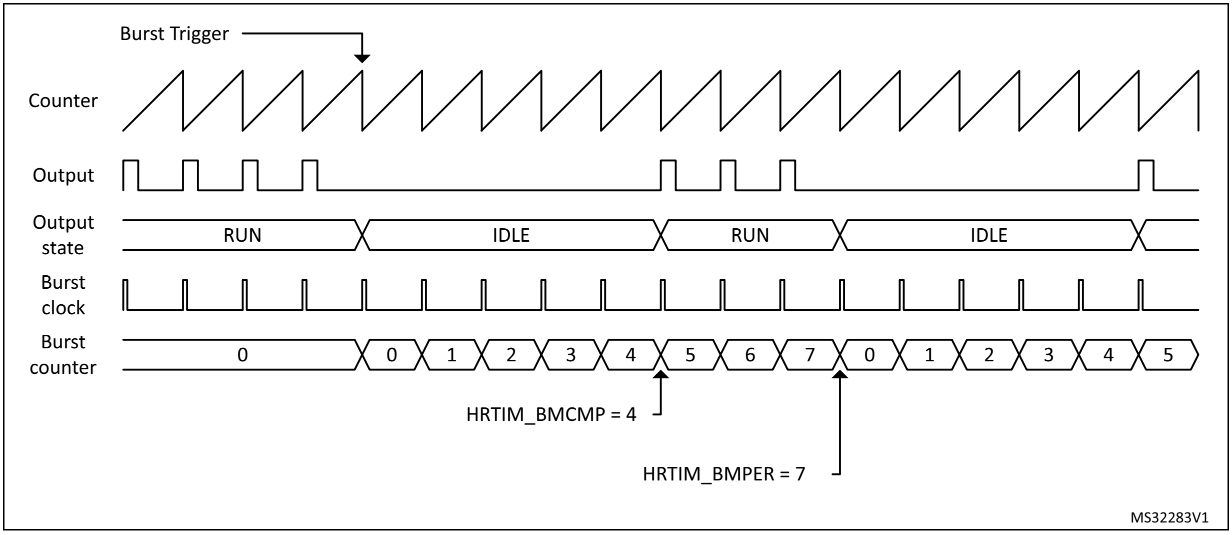
27.3.15 Burst mode controller

The burst mode controller allows to have the outputs alternatively in IDLE and RUN state, by hardware, so as to skip some switching periods with a programmable periodicity and duty cycle.

Burst mode operation is of common use in power converters when operating under light loads. It can significantly increase the efficiency of the converter by reducing the number of transitions on the outputs and the associated switching losses.

When operating in burst mode, one or a few pulses are outputs followed by an idle period equal to several counting periods, typically, where no output pulses are produced, as shown in the example on *Figure 236*.

Figure 236. Burst mode operation example



The burst mode controller consists of:

- A counter that can be clocked by various sources, either within or outside the HRTIM (typically the end of a PWM period).
- A compare register to define the number of idle periods: HRTIM_BMCMP.
- A period register to define the burst repetition rate (corresponding to the sum of the idle and run periods): HRTIM_BMPER.

The burst mode controller is able to take over the control of any of the 10 PWM outputs. The state of each output during a burst mode operation is programmed using IDLESx and IDLEMx bits in the HRTIM_OUTxR register, as in *Table 232*.

Table 232. Timer output programming for burst mode

IDLEMx	IDLESx	Output state during burst mode
0	X	No action: the output is not affected by the burst mode operation.
1	0	Output inactive during the burst
1	1	Output active during the burst

Note: IDLEMx bit must not be changed while the burst mode is active.

The burst mode controller only acts on the output stage. A number of events are still generated during the idle period:

- Output set/reset interrupt or DMA requests
- External event filtering based on Tx2 output signal
- Capture events triggered by output set/reset

During the burst mode, neither start nor reset events are generated on the hrtim_out_sync[2:1] output, even if TxBM bit is set.

Operating mode

It is necessary to have the counter enabled (TxCEN bit set) before using the burst mode on a given timing unit.The burst mode is enabled with BME bit in the HRTIM_BMCR register.

It can operate in continuous or single-shot mode, using BMOM bit in the HRTIM_BMCR register. The continuous mode is enabled when BMOM = 1. The burst operation is maintained until BMSTAT bit in HRTIM_BMCR is reset to terminate it.

In single-shot mode (BMOM = 0), the idle sequence is executed once, following the burst mode trigger, and the normal timer operation is resumed immediately after.

The duration of the idle and run periods is defined with a burst mode counter and 2 registers. The HRTIM_BMCMPR register defines the number of counts during which the selected timer(s) are in an idle state (idle period). HRTIM_BMPER defines the overall burst mode period (sum of the idle and run periods). Once the initial burst mode trigger has occurred, the idle period length is HRTIM_BMCMPR+1, the overall burst period is HRTIM_BMPER+1.

Note: The burst mode period must not be less than or equal to the deadtime duration defined with DTRx[8:0] and DTFx[8:0] bitfields.

The counters of the timing units and the master timer can be stopped and reset during the burst mode operation. HRTIM_BMCR holds 6 control bits for this purpose: MTBM (master) and TABM..TEBM for timer A..E.

When MTBM or TxBM bit is reset, the counter clock is maintained. This allows to keep a phase relationship with other timers in multiphase systems, for instance.

When MTBM or TxBM bit is set, the corresponding counter is stopped and maintained in reset state during the burst idle period. This allows to have the timer restarting a full period when exiting from idle. If SYNCSRC[1:0] = 00 or 10 (synchronization output on the master start or timer A start), a pulse is sent on the HRTIM_SCOUT output when exiting the burst mode.

Note: TxBM bit must not be set when the balanced idle mode is active (DLYPRT[1:0] = 0x11).

Burst mode clock

The burst mode controller counter can be clocked by several sources, selected with BMCLK[3:0] bits in the HRTIM_BMCR register:

- BMCLK[3:0] = 0000 to 0101: master timer and TIMA..E reset/roll-over events. This allows to have burst mode idle and run periods aligned with the timing unit counting period (both in free-running and counter reset mode).
- BMCLK[3:0] = 0110 to 1001: The clocking is provided by the hrtim_bm_ck[4:1] inputs connected to general purpose timers, as in *Table 210*. In this case, the burst mode idle and run periods are not necessarily aligned with timing unit counting period (a pulse on the output may be interrupted, resulting a waveform with modified duty cycle for instance.
- BMCLK[3:0] = 1010: The f_{HRTIM} clock prescaled by a factor defined with BMPRSC[3:0] bits in HRTIM_BMCR register. In this case, the burst mode idle and run periods are not necessarily aligned with the timing unit counting period (a pulse on the output may be interrupted, resulting in a waveform with a modified duty cycle, for instance.

The pulse width on TIMx OC output must be at least N f_{HRTIM} clock cycles long to be detected by the HRTIM burst mode controller.

Burst mode triggers

To trigger the burst operation, 32 sources are available and are selected using the HRTIM_BMTRGR register:

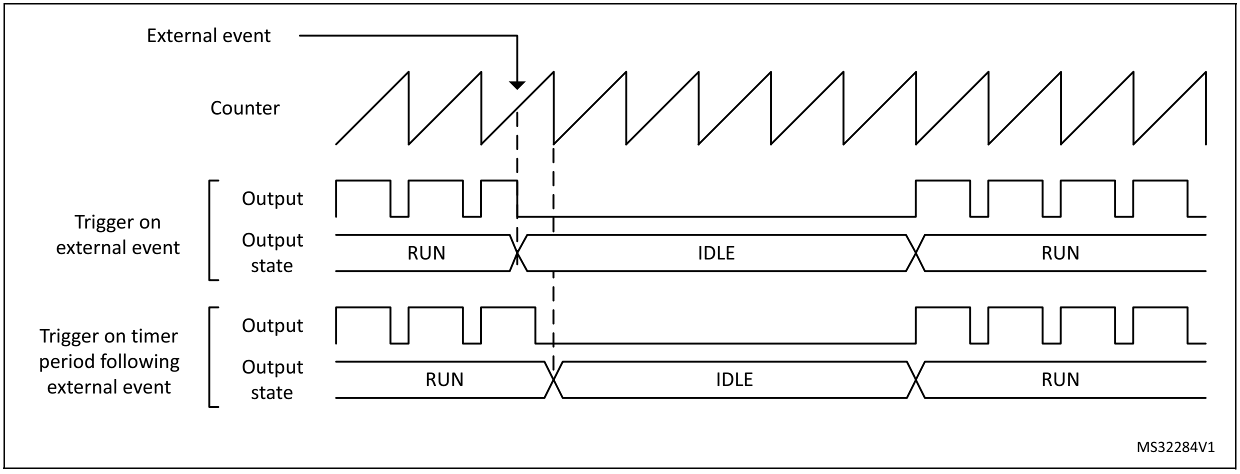
- Software trigger (set by software and reset by hardware)
- 6 master timer events: repetition, reset/roll-over, compare 1 to 4
- 5 x 4 events from timers A..F: repetition, reset/roll-over, compare 1 and 2
- External event 7 (including TIMA event filtering) and 8 (including TIMD event filtering)
- Timer A period following external event 7 (including TIMA event filtering)
- Timer D period following external event 8 (including TIMD event filtering)
- An on-chip event on the hrtim_bm_trg input (connected to the general-purpose timer TRGO output), see *Table 207* for details.

These sources can be combined to have multiple concurrent triggers.

Burst mode is not re-triggerable. In continuous mode, new triggers are ignored until the burst mode is terminated, while in single-shot mode, the triggers are ignored until the current burst completion including run periods (HRTIM_BMPER+1 cycles). This is also valid for software trigger (the software bit is reset by hardware even if it is discarded).

Figure 237 shows how the burst mode is started in response to an external event, either immediately or on the timer period following the event.

Figure 237. Burst mode trigger on external event



For TAEEV7 and TDEEV8 combined triggers (trigger on a timer period following an external event), the external event detection is always active, regardless of the burst mode programming and the on-going burst operation:

- When the burst mode is enabled (BME=1) or the trigger is enabled (TAEEV7 or TDEEV8 bit set in the BMTRG register) in between the external event and the timer period event, the burst is triggered.
- The single-shot burst mode is re-triggered even if the external event occurs before the burst end (as long as the corresponding period happens after the burst).

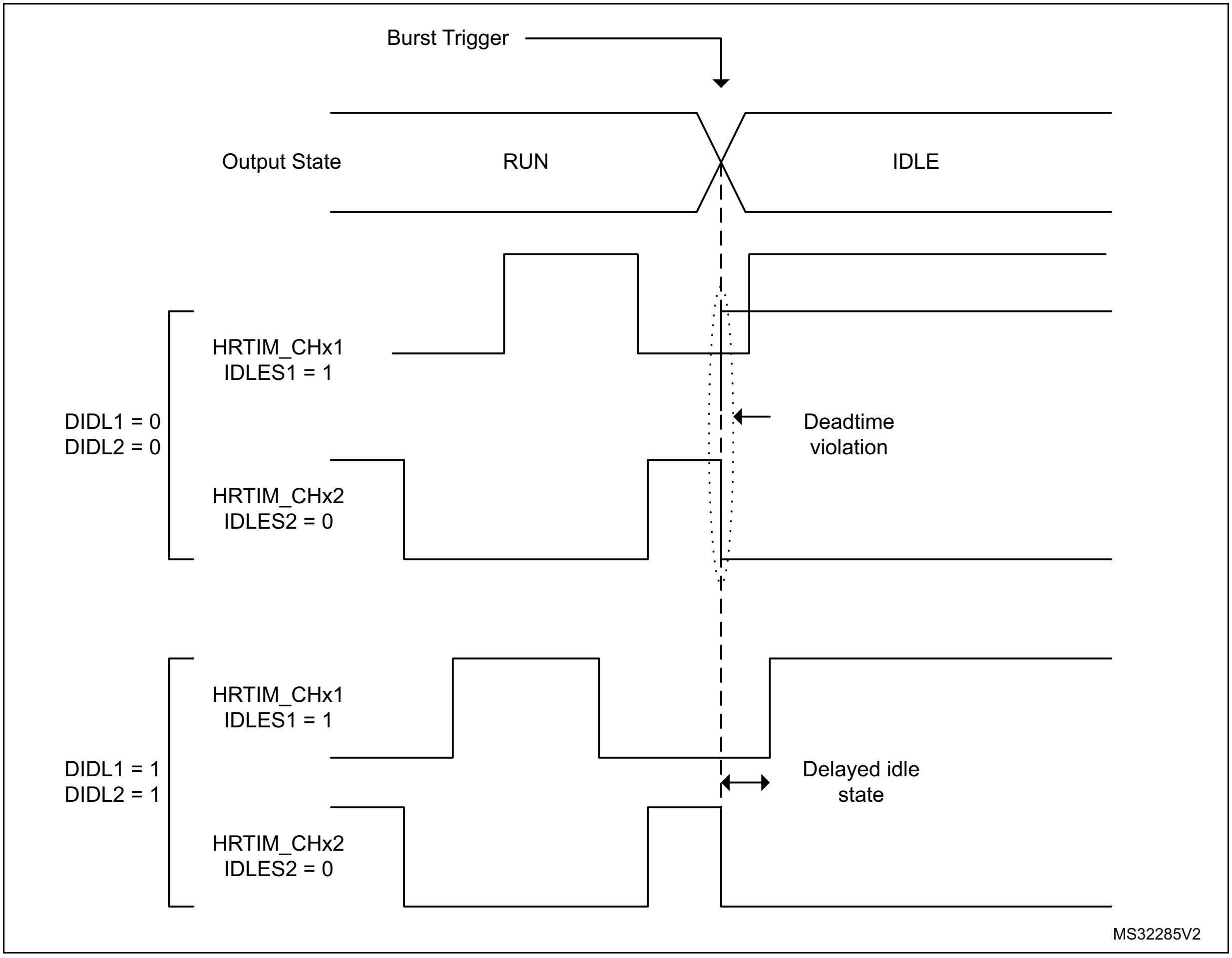
Note: TAEEV7 and TDEEV8 triggers are valid only after a period event. If the counter is reset before the period event, the pending hrtim_eev7 or hrtim_eev8 event is discarded.

Burst mode delayed entry

By default, the outputs are taking their idle level (as per IDLES1 and IDLES2 setting) immediately after the burst mode trigger.

It is also possible to delay the burst mode entry and force the output to an inactive state during a programmable period before the output takes its idle state. This is useful when driving two complementary outputs, one of them having an active idle state, to avoid a deadtime violation as shown on *Figure 238*. This prevents any risk of shoot through current in half-bridges, but causes a delayed response to the burst mode entry.

Figure 238. Delayed burst mode entry with deadtime enabled and IDLESx = 1



The delayed burst entry mode is enabled with DIDLx bit in the HRTIM_OUTxR register (one enable bit per output). It forces a deadtime insertion before the output takes its idle state. Each TIMx output has its own deadtime value:

- DTRx[8:0] on output 1 when DIDL1 = 1
- DTFx[8:0] on output 2 when DIDL2 = 1

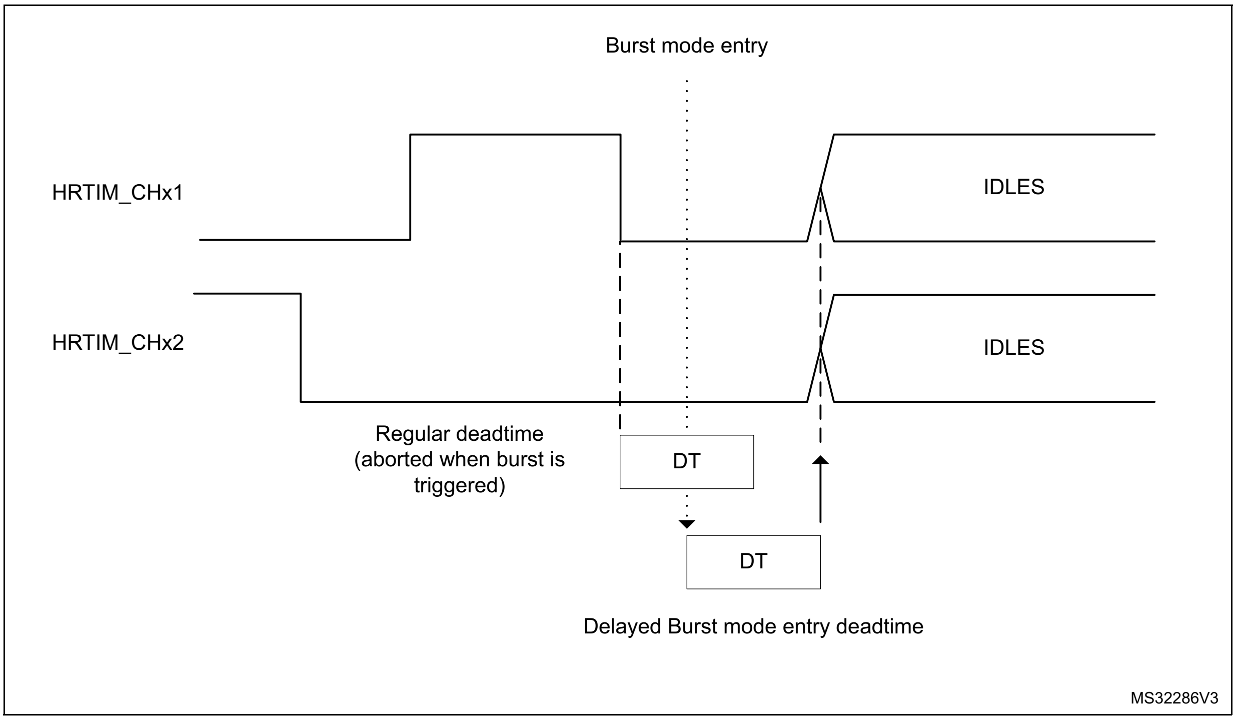
DIDLx bits can be set only if one of the outputs has an active idle level during the burst mode (IDLES = 1) and only when positive deadtimes are used (SDTR/SDTF set to 0).

Note: The delayed burst entry mode uses deadtime generator resources. Consequently, when any of the 2 DIDLx bits is set and the corresponding timing unit uses the deadtime insertion (DTEN bit set in HRTIM_OUTxR), it is not possible to use the timerx output 2 as a filter for external events (Tx2 filtering signal is not available).

When durations defined by DTRx[8:0] and DTFx[8:0] are lower than 3 f_{HRTIM} clock cycle periods, the limitations related to the narrow pulse management listed in *Section 27.3.7* must be applied.

When the burst mode entry arrives during the regular deadtime, it is aborted and a new deadtime is re-started corresponding to the inactive period, as on *Figure 239*.

Figure 239. Delayed burst mode entry during deadtime


Burst mode entry
HRTIM_CHx1
HRTIM_CHx2
IDLES
IDLES
Regular deadtime (aborted when burst is triggered)
DT
DT
Delayed Burst mode entry deadtime
MS32286V3

Burst mode exit

The burst mode exit is either forced by software (in continuous mode) or once the idle period is elapsed (in single-shot mode). In both cases, the counter is re-started immediately (if it was hold in a reset state with MTBM or TxBM bit = 1), but the effective output state transition from the idle to active mode only happens after the programmed set/reset event.

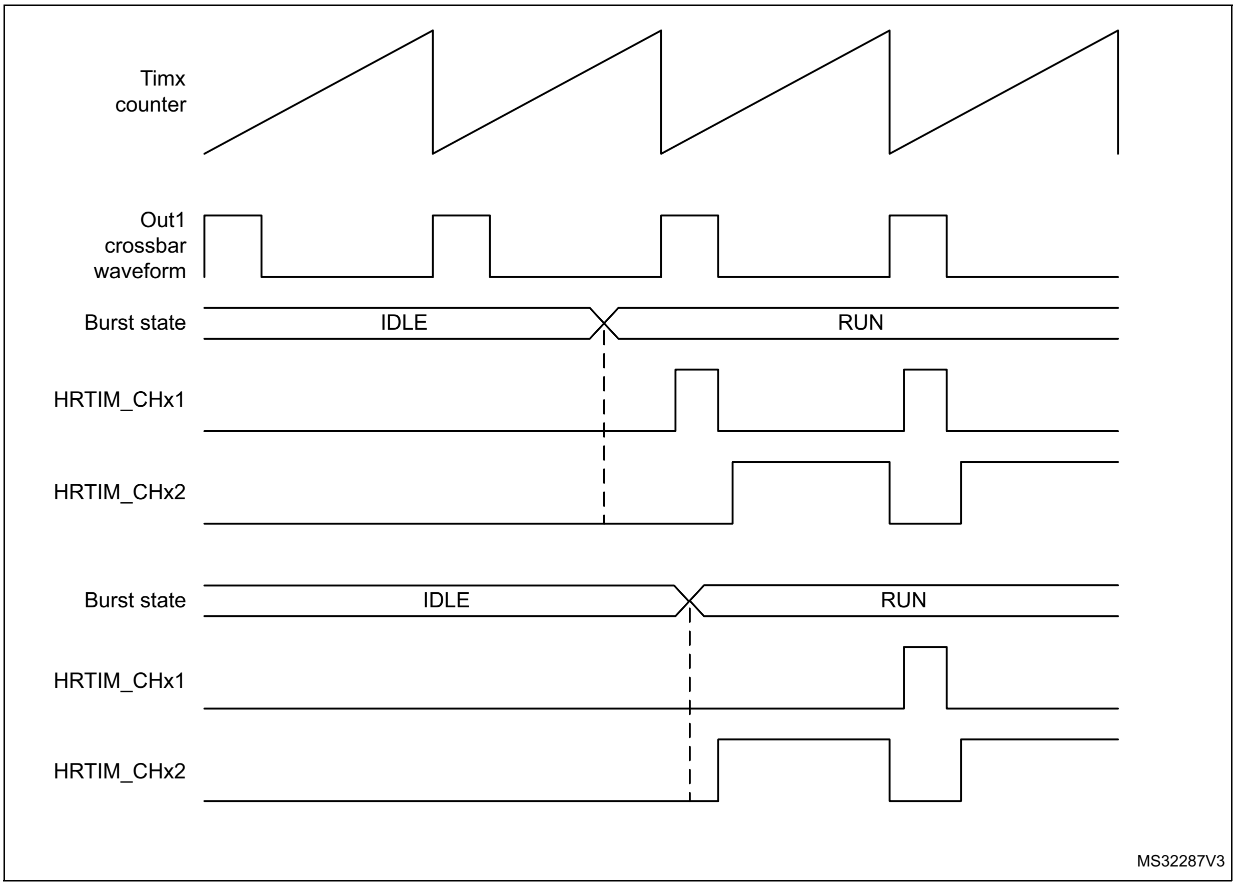
A burst period interrupt is generated in single-shot and continuous modes when BMPERIE enable bit is set in the HRTIM_IER register. This interrupt can be used to synchronize the burst mode exit with a burst period in continuous burst mode.

Figure 240 shows how a normal operation is resumed when the deadtime is enabled. Although the burst mode exit is immediate, this is only effective on the first set event on any of the complementary outputs.

Two different cases are presented:

1. The burst mode ends while the signal is inactive on the crossbar output waveform. The active state is resumed on Tx1 and Tx2 on the set event for the Tx1 output, and the Tx2 output does not take the complementary level on burst exit.
2. The burst mode ends while the crossbar output waveform is active: the activity is resumed on the set event of Tx2 output, and Tx1 does not take the active level immediately on burst exit.

Figure 240. Burst mode exit when the deadtime generator is enabled



The behavior described above is slightly different when the push-pull mode is enabled. The push-pull mode forces an output reset at the beginning of the period if the output is inactive, or symmetrically forces an active level if the output was high during the preceding period.

Consequently, an output with an active idle state can be reset at the time the burst mode is exited even if no transition is explicitly programmed. For symmetrical reasons, an output can be set at the time the burst mode is exited even if no transition is explicitly programmed, in case it was active when it entered in idle state.

Burst mode registers preloading and update

BMPREN bit (burst mode preload enable) allows to have the burst mode compare and period registers preloaded (HRTIM_BMCMP and HRTIM_BMPER).

When BMPREN is set, the transfer from preload to active register happens:

- when the burst mode is enabled (BME = 1),
- at the end of the burst mode period.

A write into the HRTIM_BMPER period register disables the update temporarily, until the HRTIM_BMCMP compare register is written, to ensure the consistency of the two registers when they are modified.

If the compare register only needs to be changed, a single write is necessary. If the period only needs to be changed, it is also necessary to re-write the compare to have the new values taken into account.

When BMPREN bits is reset, the write access into BMCMPR and BMPER directly updates the active register. In this case, it is necessary to consider when the update is done during the overall burst period, for the 2 cases below:

a) Compare register update

If the new compare value is above the current burst mode counter value, the new compare is taken into account in the current period.

If the new compare value is below the current burst mode counter value, the new compare is taken into account in the next burst period in continuous mode, and ignored in single-shot mode (no compare match occurs and the idle state lasts until the end of the idle period).

b) Period register update

If the new period value is above the current burst mode counter value, the change is taken into account in the current period.

Note: If the new period value is below the current burst mode counter value, the new period is not taken into account, the burst mode counter overflows (at 0xFFFF) and the change is effective in the next period. In single-shot mode, the counter rolls over at 0xFFFF and the burst mode re-starts for another period up to the new programmed value.

Burst mode emulation using a compound register

The burst mode controller only controls one or a set of timers for a single converter. When the burst mode is necessary for multiple independent timers, it is possible to emulate a simple burst mode controller using the DMA and the HRTIM_CMP1CxR compound register, which holds aliases of both the repetition and the compare 1 registers.

This is applicable to a converter which only requires a simple PWM (typically a buck converter), where the duty cycle only needs to be updated. In this case, the CMP1 register is used to reset the output (and define the duty cycle), while it is set on the period event.

In this case, a single 32-bit write access in CMP1CxR is sufficient to define the duty cycle (with the CMP1 value) and the number of periods during which this duty cycle is maintained (with the repetition value). To implement a burst mode, it is then only necessary to transfer by DMA (upon repetition event) two 32-bit data in continuous mode, organized as follows:

CMPC1xR = {REP_Run; CMP1 = Duty_Cycle}, {REP_Idle; CMP1 = 0}

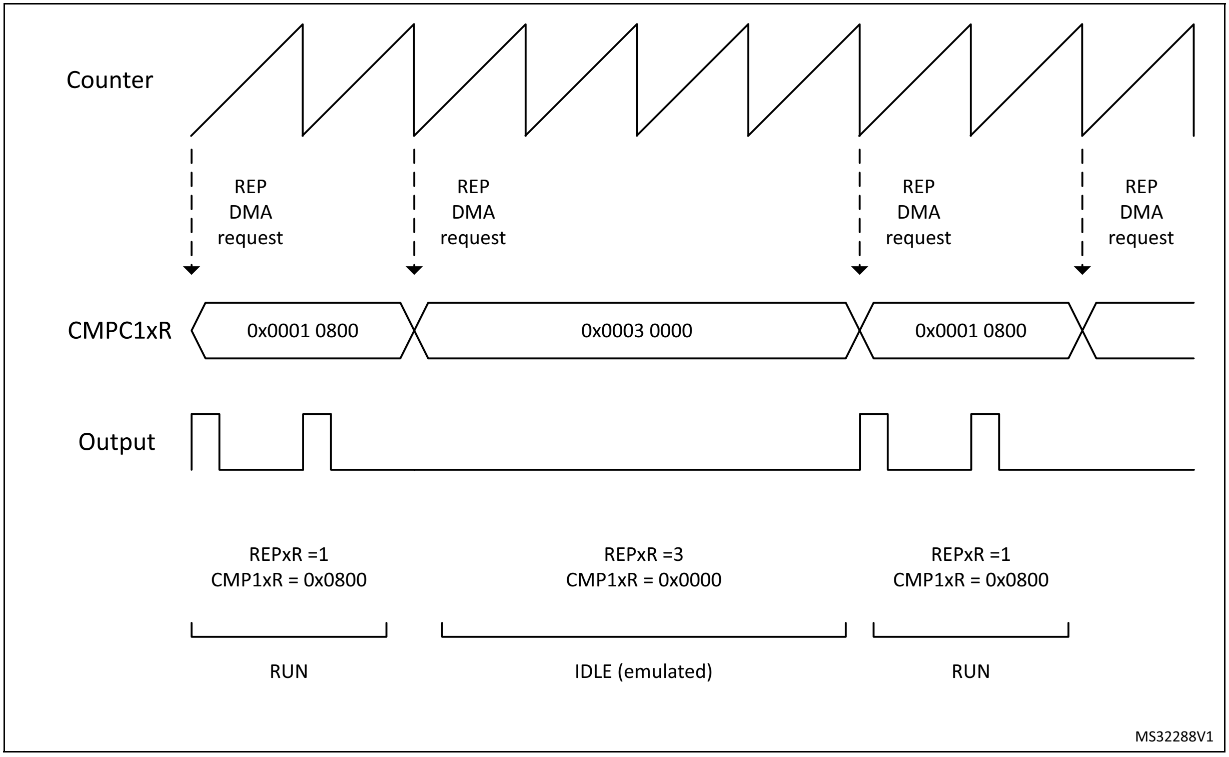
For instance, the values:

{0x0003 0000}: CMP1 = 0 for 3 periods

{0x0001 0800}: CMP1 = 0x0800 for 1 period

provide a burst mode with 2 periods active every 6 PWM periods, as shown on Figure 241.

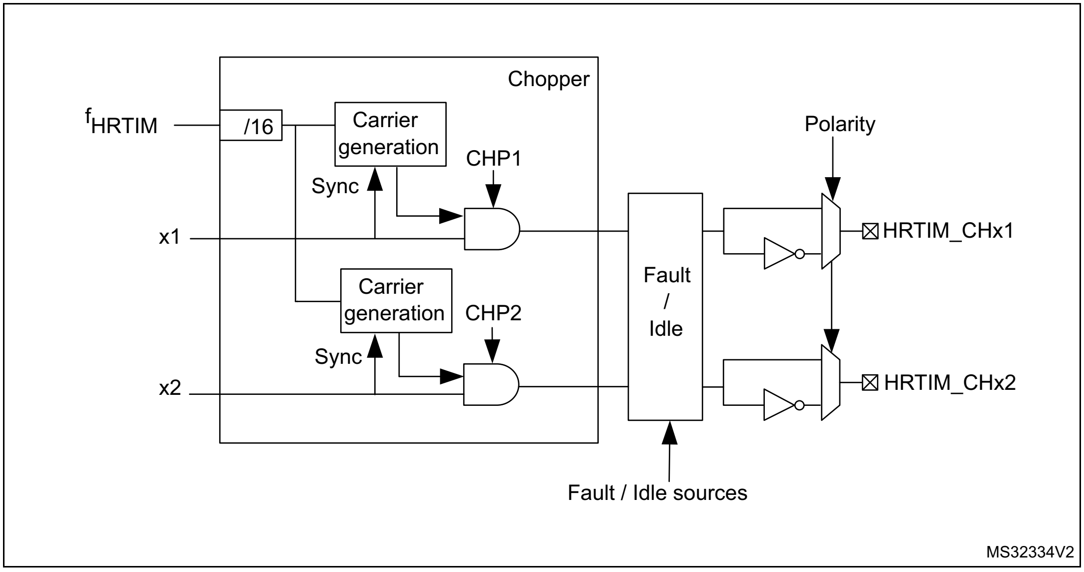
Figure 241. Burst mode emulation example



27.3.16 Chopper

A high-frequency carrier can be added on top of the timing unit output signals to drive isolation transformers. This is done in the output stage before the polarity insertion, as shown on *Figure 242*, using CHP1 and CHP2 bits in the HRTIM_OUTxR register, to enable chopper on outputs 1 and 2, respectively.

Figure 242. Carrier frequency signal insertion



The chopper parameters can be adjusted using the HRIM_CHPxR register, with the possibility to define a specific pulsewidth at the beginning of the pulse, to be followed by a carrier frequency with programmable frequency and duty cycle, as in *Figure 243*.

CARFRQ[3:0] bits define the frequency, ranging from 664.06 kHz to 10.625 MHz (for f_{HRTIM} = 170 MHz) following the formula $F_{CHPFRQ} = f_{HRTIM} / (16 \times (CARFRQ[3:0]+1))$.

The duty cycle can be adjusted by 1/8 step with CARDTY[2:0], from 0/8 up to 7/8 duty cycle. When CARDTY[2:0] = 000 (duty cycle = 0/8), the output waveform only contains the starting pulse following the rising edge of the reference waveform, without any added carrier.

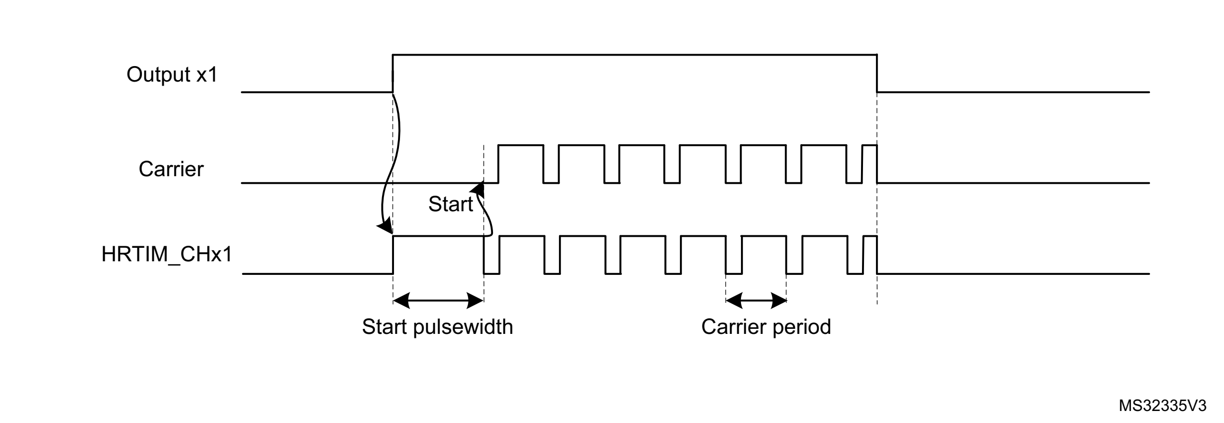
The pulsewidth of the initial pulse is defined using the STRPW[3:0] bitfield as follows: $t1STPW = (STRPW[3:0]+1) \times 16 \times t_{HRTIM}$ and ranges from 94 ns to 1.51 µs (for f_{HRTIM}=170 MHz).

The carrier frequency parameters are defined based on the f_{HRTIM} frequency, and are not dependent from the CKPSC[2:0] setting.

In chopper mode, the carrier frequency and the initial pulsewidth are combined with the reference waveform using an AND function. A synchronization is performed at the end of the initial pulse to have a repetitive signal shape.

The chopping signal is stopped at the end of the output waveform active state, without waiting for the current carrier period to be completed. It can thus contain shorter pulses than programmed.

Figure 243. HRTIM outputs with Chopper mode enabled



Note: CHP1 and CHP2 bits must be set prior to the output enable done with TxyOEN bits in the HRTIM_OENR register.

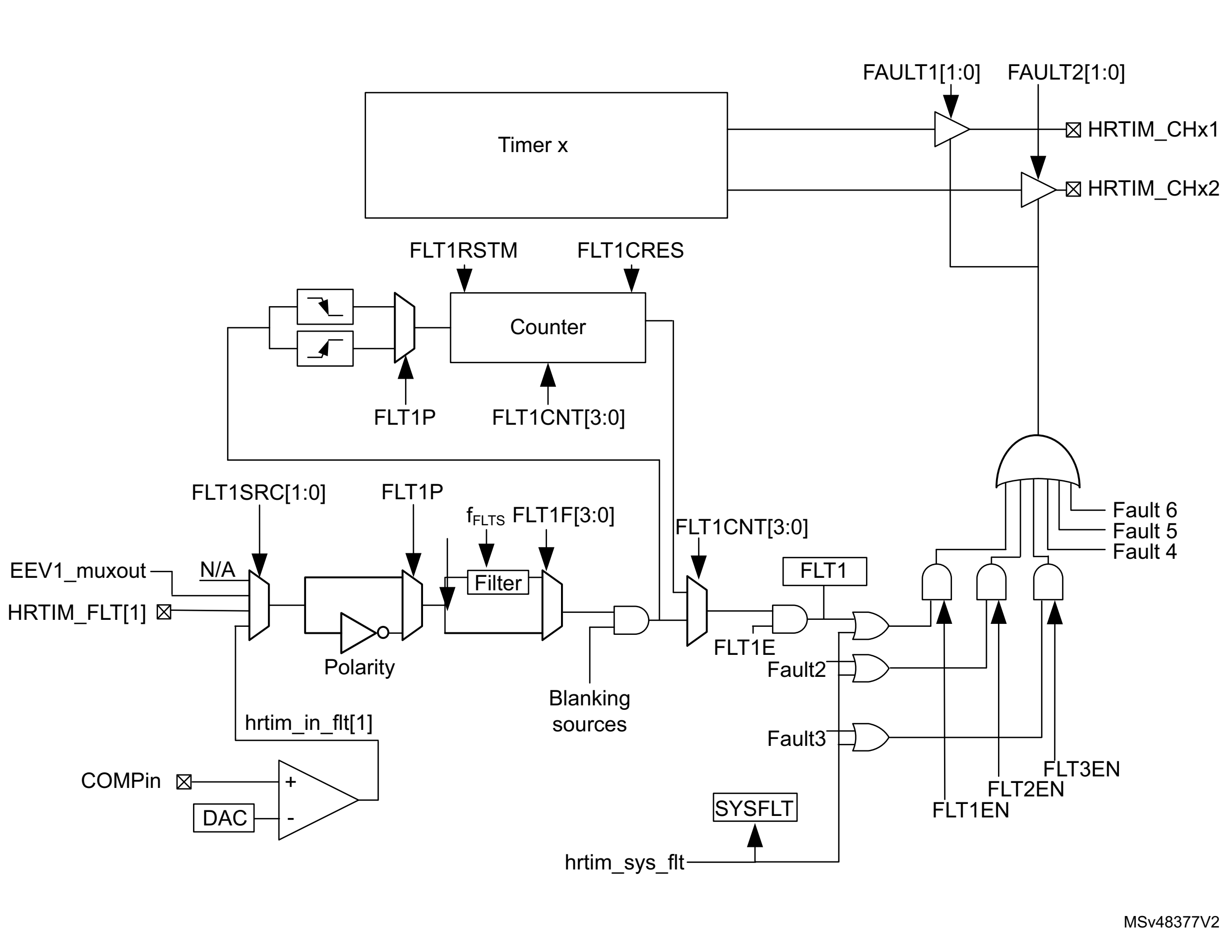
CARFRQ[2:0], CARDTY[2:0] and STRPW[3:0] bitfields cannot be modified while the chopper mode is active (at least one of the two CHPx bits is set).

27.3.17 Fault protection

The HRTIM has a versatile fault protection circuitry to disable the outputs in case of an abnormal operation. Once a fault has been triggered, the outputs take a predefined safe state. This state is maintained until the output is re-enabled by software. In case of a permanent fault request, the output remains in its fault state, even if the software attempts to re-enable them, until the fault source disappears.

The HRTIM has 6 FAULT input channels; all of them are available and can be combined for each of the 6 timing units, as shown on *Figure 244*.

Figure 244. Fault protection circuitry (FAULT1 fully represented, FAULT2..6 partially)



Each fault channel is fully configurable using HRTIM_FLTINR1 and HRTIM_FLTINR2 registers before being routed to the timing units. FLTxSRC FLTxSRC[1:0] bit selects the source of the fault signal, that can be either a digital input or an internal event (built-in comparator output).

Table 233 and *Table 211* summarize the available sources for each of the 6 faults channels:

Table 233. Fault inputs

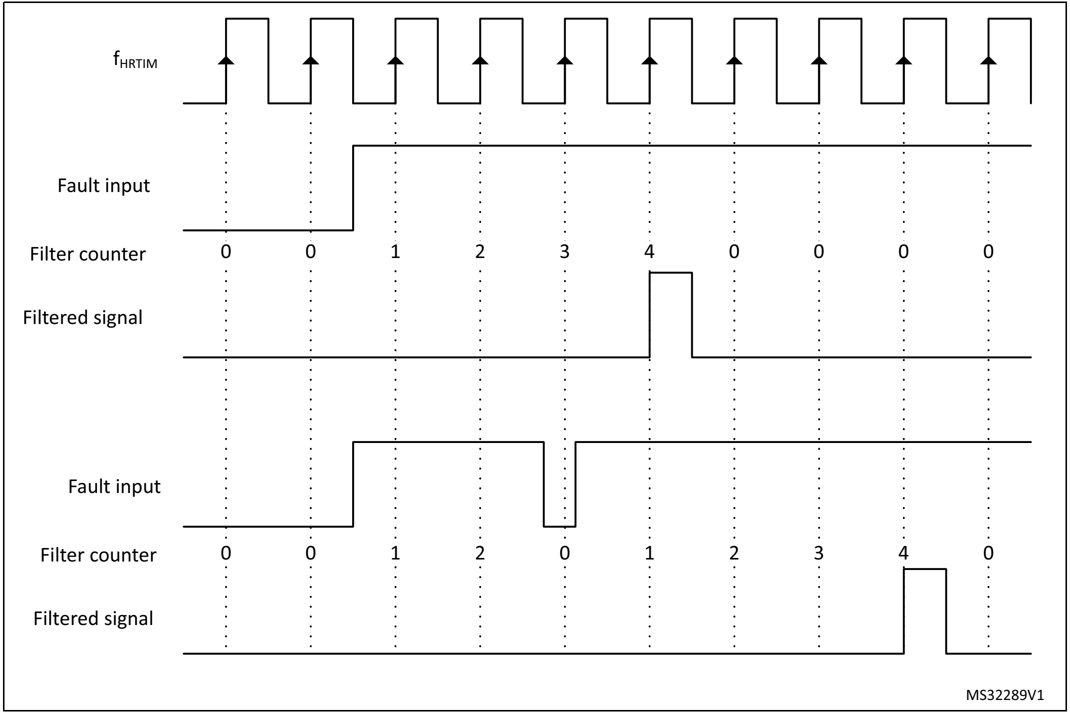
Fault channel	External Input FLTxSRC[1:0] = 00	On-chip source FLTxSRC[1:0] = 01	External Input FLTxSRC[1:0] = 10	On-chip source FLTxSRC[1:0] = 11
hrtim_flt1[4:1]	HRTIM_FLT1	Comparator output	EEV1_muxout	N/A
hrtim_flt2[4:1]	HRTIM_FLT2	Comparator output	EEV2_muxout	N/A
hrtim_flt3[4:1]	HRTIM_FLT3	Comparator output	EEV3_muxout	N/A
hrtim_flt4[4:1]	HRTIM_FLT4	Comparator output	EEV4_muxout	N/A
hrtim_flt5[4:1]	HRTIM_FLT5	Comparator output	EEV5_muxout	N/A
hrtim_flt6[4:1]	HRTIM_FLT6	Comparator output	EEV6_muxout	N/A

The EEVx_muxout event mentioned in *Table 233* above is taken after the hrtim_eevx[4:1] input multiplexer controlled by the EExSRC[1:0]bits. Refer to *Figure 216* for details.

The polarity of the signal can be selected to define the active level, using the FLTxP polarity bit in HRTIM_FLTINRx registers. If FLTxP = 0, the signal is active at low level; if FLTxP = 1, it is active when high.

The fault information can be filtered after the polarity setting. If FLTxF[3:0] bitfield is set to 0000, the signal is not filtered and acts asynchronously, independently from the f_{HRTIM} clock. For all other FLTxF[3:0] bitfield values, the signal is digitally filtered. The digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected). *Figure 245* shows how a spurious fault signal is filtered.

Figure 245. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4)



The filtering period ranges from 2 cycles of the f_{HRTIM} clock up to 8 cycles of the f_{FLTS} clock divided by 32. f_{FLTS} is defined using FLTSD[1:0] bits in the HRTIM_FLTINR2 register. *Table 234* summarizes the sampling rate and the filter length. A jitter of 1 sampling clock period must be subtracted from the filter length to take into account the uncertainty due to the sampling and have the effective filtering.

Table 234. Sampling rate and filter length vs FLTFxF[3:0] and clock setting

-	f_{FLTS} vs FLTSD[1:0]				Filter length for f_{HRTIM} = 170 MHz	
FLTFxF[3:0]	00	01	10	11	Min	Max
0001,0010,0011	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}, N =2 11.8 ns	f_{HRTIM}, N =8 47.1 ns
0100, 0101	f_{HRTIM} /2	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /2, N = 6 70.6 ns	f_{HRTIM} /16, N = 8 753 ns
0110, 0111	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /4, N = 6 141 ns	f_{HRTIM} /32, N = 8 1.51 µs
1000, 1001	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /8, N = 6 282 ns	f_{HRTIM} /64, N = 8 3.01 µs
1010, 1011, 1100	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /16, N = 5 471 ns	f_{HRTIM} /128, N = 8 6.02 µs
1101, 1110, 1111	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /256	f_{HRTIM} /32, N = 5 941 ns	f_{HRTIM} /256, N = 8 12.05 µs

Fault blanking and event counting

The fault inputs can be temporary disabled to blank spurious fault events. The blanking sources are listed in the *Table 235* below.

Table 235. Fault input blanking events

-	FLTxBLKS = 0, reset-aligned window		FLTxBLKS = 1 moving window	
Fault input	Blanking window start	Blanking window end	Blanking window start	Blanking window end
hrtim_flt1[4:1]	Timer A reset/roll-over	Timer A CMP3 event	Timer A CMP4 event	Timer A CMP3 event
hrtim_flt2[4:1]	Timer B reset/roll-over	Timer B CMP3 event	Timer B CMP4 event	Timer B CMP3 event
hrtim_flt3[4:1]	Timer C reset/roll-over	Timer C CMP3 event	Timer C CMP4 event	Timer C CMP3 event
hrtim_flt4[4:1]	Timer D reset/roll-over	Timer D CMP3 event	Timer D CMP4 event	Timer D CMP3 event
hrtim_flt5[4:1]	Timer E reset/roll-over	Timer E CMP3 event	Timer E CMP4 event	Timer E CMP3 event
hrtim_flt6[4:1]	Timer F reset/roll-over	Timer F CMP3 event	Timer F CMP4 event	Timer F CMP3 event

A fault counter also allows to discard multiple spurious fault events and define an acceptance criteria.

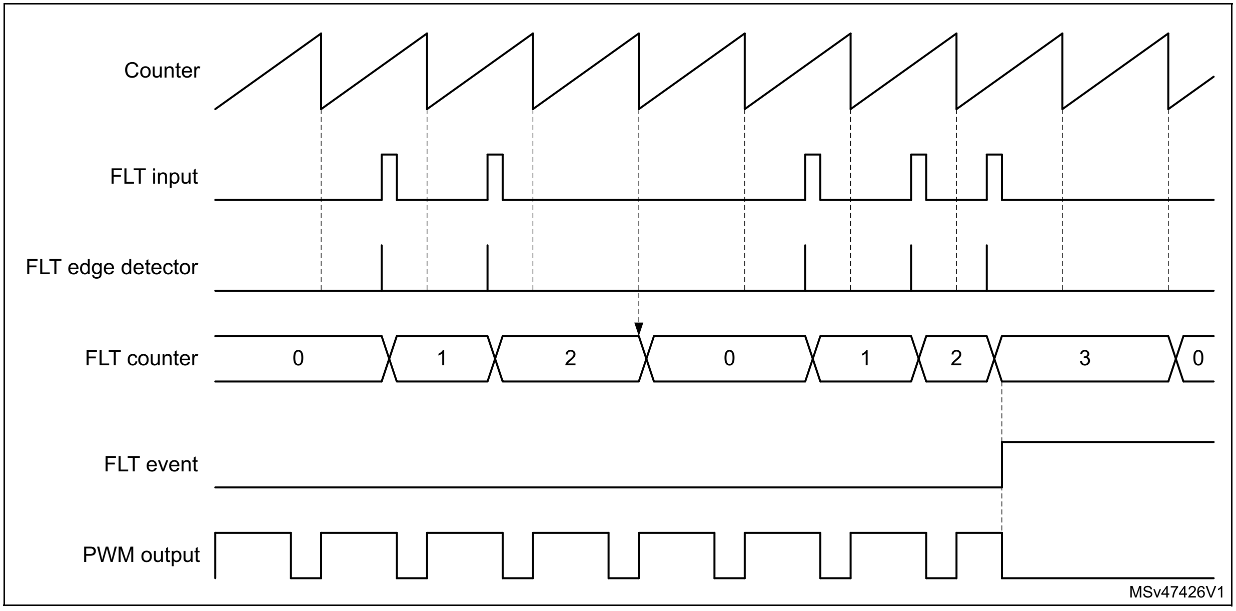
The FLTxCNT[3:0] bitfield selects the FAULTx counter threshold. A fault is considered valid when the number of events is equal to the FLTxCNT[3:0] value.

This FLTxRSTM selects the FAULTx counter reset mode

- 0: fault counter hardware reset on reset / roll-over event, as per the *Table 236* below.
- 1: fault counter reset on each reset / roll-over event only if no event occurs during last counting period, as shown on *Figure 246* below.

The fault counter can be reset by software with the FLTxCRES bit at anytime.

Figure 246. Fault counter cumulative mode (FLTxRSTM = 1, FLTxCNT[3:0] = 2)



A given FLTx input counter can be reset by a single source. The *Table 236* indicates which timer unit associated with a given fault. This does not prevent to have a fault line shared by multiple timer (e.g. FLT1 with event counter enabled, acting on timer A, timer B and timer C simultaneously).

Table 236. Fault 1..6 counter reset source

Fault Input	Fault counter reset source
hrtim_flt1[4:1]	Timer A reset/roll-over
hrtim_flt2[4:1]	Timer B reset/roll-over
hrtim_flt3[4:1]	Timer C reset/roll-over
hrtim_flt4[4:1]	Timer D reset/roll-over
hrtim_flt5[4:1]	Timer E reset/roll-over
hrtim_flt6[4:1]	Timer F reset/roll-over

System fault input (hrtim_sys_flt)

This fault is provided by the MCU Class B circuitry (see the system configuration controller (SYSCFG) section for details) and corresponds to a system fault coming from:

- the clock security system
- the SRAM parity checker
- the Cortex®-M4 with FPU lockup signal
- the PVD detector
- the Flash ECC double error detection

This input overrides the FAULT inputs and disables all outputs having FAULTy[1:0] = 01, 10, 11.

For each FAULT channel, a write-once FLTxLCK bit in the HRTIM_FLTxR register allows to lock FLTxE, FLTxP, FLTxSRC, FLTxF[3:0] bits (it renders them read-only), for functional safety purpose. If enabled, the fault conditioning set-up is frozen until the next HRTIM or system reset.

Once the fault signal is conditioned as explained above, it is routed to the timing units. For any of them, the 6 fault channels are enabled using bits FLT1EN to FLT6EN in the HRTIM_FLTxR register, and they can be selected simultaneously (the sysfault is automatically enabled as long as the output is protected by the fault mechanism). This allows to have, for instance:

- One fault channel simultaneously disabling several timing units
- Multiple fault channels being ORed to disable a single timing unit

A write-once FLTLCK bit in the HRTIM_FLTxR register allows to lock FLTxEN bits (it renders them read-only) until the next reset, for functional safety purpose. If enabled, the timing unit fault-related set-up is frozen until the next HRTIM or system reset.

For each of the timers, the output state during a fault is defined with FAULT1[1:0] and FAULT2[1:0] bits in the HRTIM_OUTxR register (see *Section 27.3.14*).

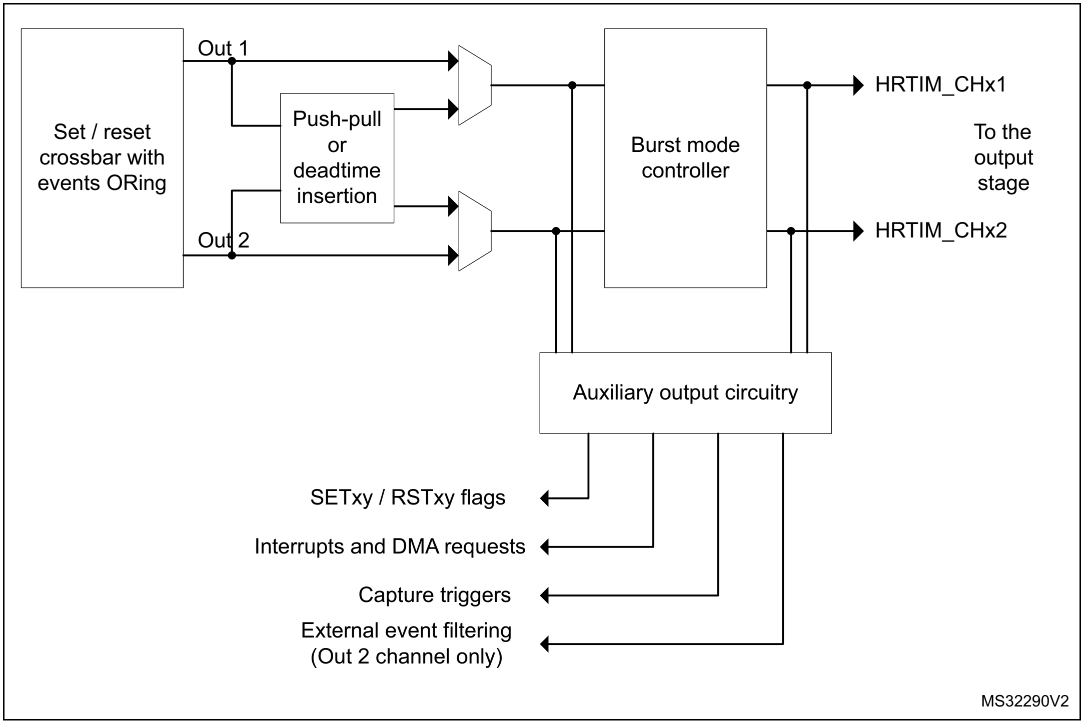
27.3.18 Auxiliary outputs

Timer A to E have auxiliary outputs in parallel with the regular outputs going to the output stage. They provide the following internal status, events and signals:

- SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests
- Capture triggers upon output set/reset
- External event filters following a Tx2 output copy (see details in *Section 27.3.9*)

The auxiliary outputs are taken either before or after the burst mode controller, depending on the HRTIM operating mode. An overview is given on *Figure 247*.

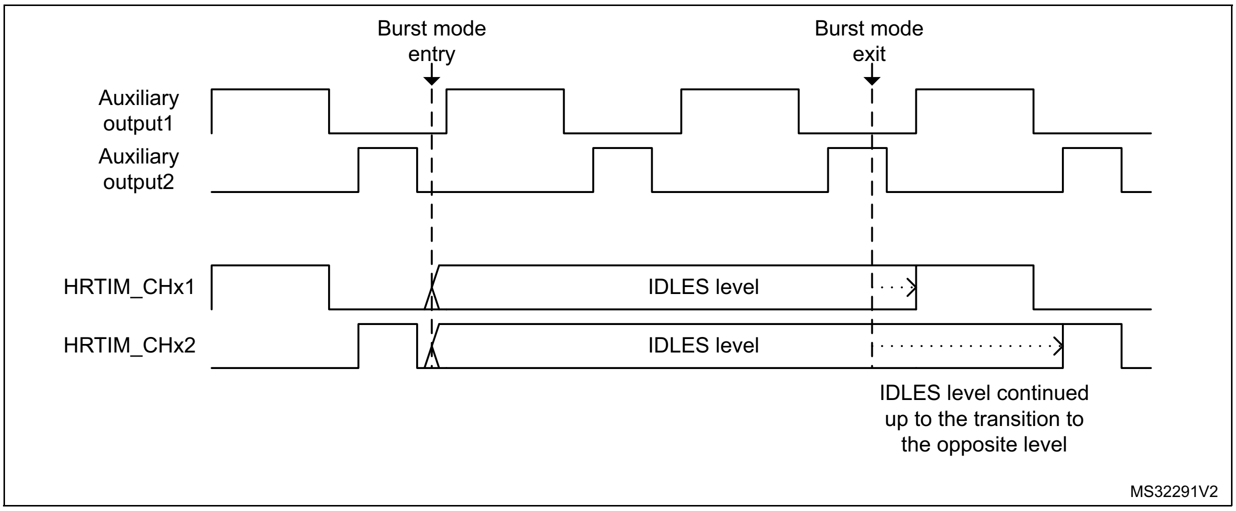
Figure 247. Auxiliary outputs


Set / reset crossbar with events ORing
Out 1
Out 2
Push-pull or deadtime insertion
Burst mode controller
HRTIM_CHx1
HRTIM_CHx2
To the output stage
Auxiliary output circuitry
SETxy / RSTxy flags
Interrupts and DMA requests
Capture triggers
External event filtering (Out 2 channel only)
MS32290V2

By default, the auxiliary outputs are copies of outputs Tx1 and Tx2. The exceptions are:

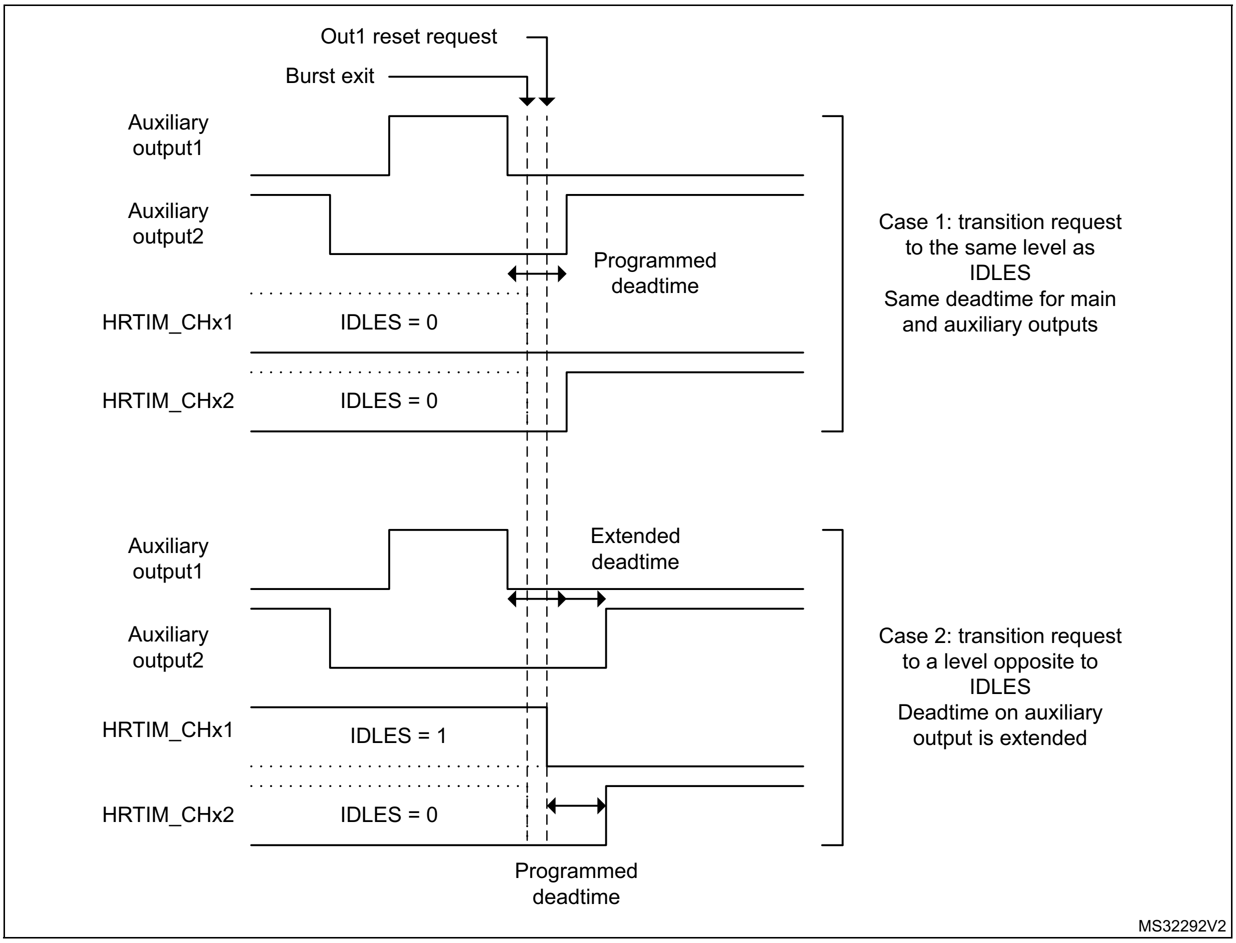
- The delayed idle and the balanced idle protections, when the deadtime is disabled (DTEN = 0). When the protection is triggered, the auxiliary outputs are maintained and follow the signal coming out of the crossbar. On the contrary, if the deadtime is enabled (DTEN = 1), both main and auxiliary outputs are forced to an inactive level.
- The burst mode (TCEN=1, IDLEMx=1); there are 2 cases:
  - a) If DTEN=0 or DIDLx=0, the auxiliary outputs are not affected by the burst mode entry and continue to follow the reference signal coming out of the crossbar (see *Figure 248*).
  - b) If the deadtime is enabled (DTEN=1) together with the delayed burst mode entry (DIDLx=1), the auxiliary outputs have the same behavior as the main outputs. They are forced to the IDLES level after a deadtime duration, then they keep this level during all the burst period. When the burst mode is terminated, the IDLES level is maintained until a transition occurs to the opposite level, similarly to the main output.

Figure 248. Auxiliary and main outputs during burst mode (DIDLx = 0)



The signal on the auxiliary output can be slightly distorted when exiting from the burst mode or when re-enabling the outputs after a delayed protection, if this happens during a deadtime. In this case, the deadtime applied to the auxiliary outputs is extended so that the deadtime on the main outputs is respected. *Figure 249* gives some examples.

Figure 249. Deadtime distortion on auxiliary output when exiting burst mode



27.3.19 Synchronizing the HRTIM with other timers or HRTIM instances

The HRTIM provides options for synchronizing multiple HRTIM instances, as a master unit (generating a synchronization signal) or as a slave (waiting for a trigger to be synchronized). This feature can also be used to synchronize the HRTIM with other timers, either external or on-chip. The synchronization circuitry is controlled inside the master timer.

Synchronization output

This section explains how the HRTIM must be configured to synchronize external resources and act as a master unit.

Four events can be selected as the source to be sent to the synchronization output. This is done using SYNCSRC[1:0] bits in the HRTIM_MCR register, as follows:

- 00: master timer start
  This event is generated when MCEN bit is set or when the timer is re-started after having reached the period value in single-shot mode. It is also generated on a reset which occurs during the counting (when CONT or RETRIG bits are set).
- 01: master timer compare 1 event
- 10: timer A start
  This event is generated when TACEN bit is set or when the counter is reset and re-starts counting in response to this reset. The following counter reset events are not propagated to the synchronization output: counter roll-over in continuous mode, and discarded reset request in single-shot non-retriggerable mode. The reset is only taken into account when it occurs during the counting (CONT or RETRIG bits are set).
- 11: timer A compare 1 event

SYNCOUT[1:0] bits in the HRTIM_MCR register specify how the synchronization event is generated.

The synchronization pulses are generated on the HRTIM_SCOUT output pin, with SYNCOUT[1:0] = 1x. SYNCOUT[0] bit specifies the polarity of the synchronization signal. If SYNCOUT[0] = 0, the HRTIM_SCOUT pin has a low idle level and issues a positive pulse of 16 f_{HRTIM} clock cycles length for the synchronization). If SYNCOUT[0] = 1, the idle level is high and a negative pulse is generated.

Note: The synchronization pulse is followed by an idle level of 16 f_{HRTIM} *clock cycles during which any new synchronization request is discarded. Consequently, the maximum synchronization frequency is* f_{HRTIM}*/32.*

The idle level on the HRTIM_SCOUT pin is applied as soon as the SYNCOUT[1:0] bits are enabled (i.e. the bitfield value is different from 00).

The synchronization output initialization procedure must be done prior to the configuration of the MCU outputs and counter enable, in the following order:

1. SYNCOUT[1:0] and SYNCSRC[1:0] bitfield configuration in HRTIM_MCR
2. HRTIM_SCOUT pin configuration (see the General-purpose I/Os section)
3. Master or timer A counter enable (MCEN or TACEN bit set)

When the synchronization input mode is enabled and starts the counter (using SYNCSTRTM/SYNCSTRTx bits) simultaneously with the synchronization output mode (SYNCSRC[1:0] = 00 or 10), the output pulse is generated only when the counter is starting or is reset while running. Any reset request clearing the counter without causing it to start does not affect the synchronization output.

Synchronization input

The HRTIM can be synchronized by external sources, as per the programming of the SYNCIN[1:0] bits in the HRTIM_MCR register:

- 00: synchronization input is disabled
- 01: reserved configuration
- 10: the On-chip timer TRGO output connected to hrtim_in_sync[2] input (refer to Table 207 for details).
- 11: a positive pulse on the HRTIM_SCIN input pin (hrtim_in_sync[3])

This bitfield cannot be changed once the destination timer (master timer or timing unit) is enabled (MCEN and/or TxCEN bit set).

The HRTIM_SCIN input is rising-edge sensitive. The timer behavior is defined with the following bits present in HRTIM_MCR and HRTIM_TIMxCR registers (see Table 237 for details):

- Synchronous start: the incoming signal starts the timer's counter (SYNCSTRTM and/or SYNCSTRTx bits set). TxCEN (MCEN) bits must be set to have the timer enabled and the counter ready to start. In continuous mode, the counter does not start until the synchronization signal is received.
- Synchronous reset: the incoming signal resets the counter (SYNCRSTM and/or SYNCRSTx bits set). This event decrements the repetition counter as any other reset event.

The synchronization events are taken into account only once the related counters are enabled (MCEN or TxCEN bit set). A synchronization request triggers a SYNC interrupt.

Note: A synchronized start event resets the counter if the current counter value is above the active period value.

The effect of the synchronization event depends on the timer operating mode, as summarized in Table 237.

Table 237. Effect of sync event versus timer operating modes

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot non-retriggerable	0	1	Start events are taken into account when the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be re-started with the synchronization). Any reset occurring during the counting is ignored (as during regular non-retriggerable mode).
	1	X	Reset events are starting the timer counting. They are taken into account only if the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. When multiple reset requests are selected (from HRTIM_SCIN and from internal events), only the first arriving request is taken into account.

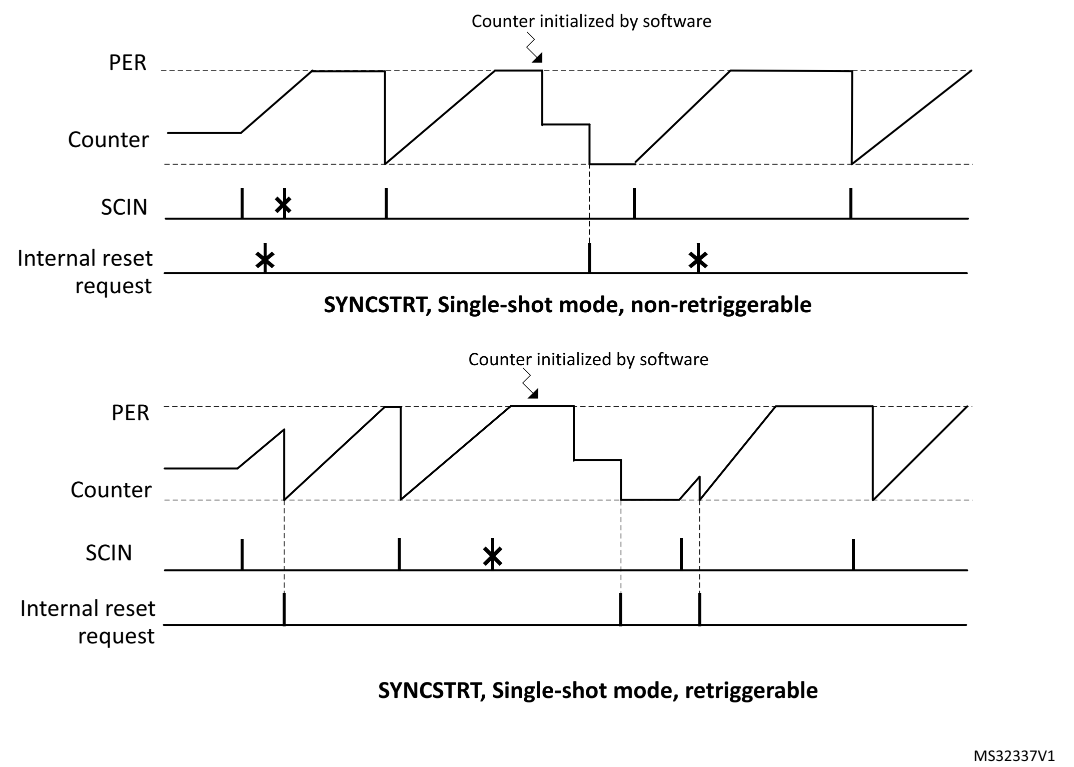
Table 237. Effect of sync event versus timer operating modes (continued)

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot retriggerable	0	1	The counter start is effective only if the counter is not started or period is elapsed. Any synchronization event occurring after counter start has no effect. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be started by the synchronization). A reset occurring during counting is taken into account (as during regular retriggerable mode).
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM's timer counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.
Continuous mode	0	1	The timer is enabled (MCEN or TxCEN bit set) and is waiting for the synchronization event to start the counter. Any synchronization event occurring after the counter start has no effect (the counter can solely be started by the synchronization). A reset request clears the counter but does not start it.
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM's timer counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.

When a synchronization reset event occurs within the same f_{HRTIM} *clock cycle as the period event, this reset is postponed to a programmed period event (since both events are causing a counter roll-over). This applies only when the high-resolution is active (*CKPSC[2:0] < 5).

Figure 250 presents how the synchronized start is done in single-shot mode.

Figure 250. Counter behavior in synchronized start mode


Counter initialized by software
PER
Counter
SCIN
Internal reset request
SYNCSTRT, Single-shot mode, non-retriggerable
Counter initialized by software
PER
Counter
SCIN
Internal reset request
SYNCSTRT, Single-shot mode, retriggerable
MS32337V1

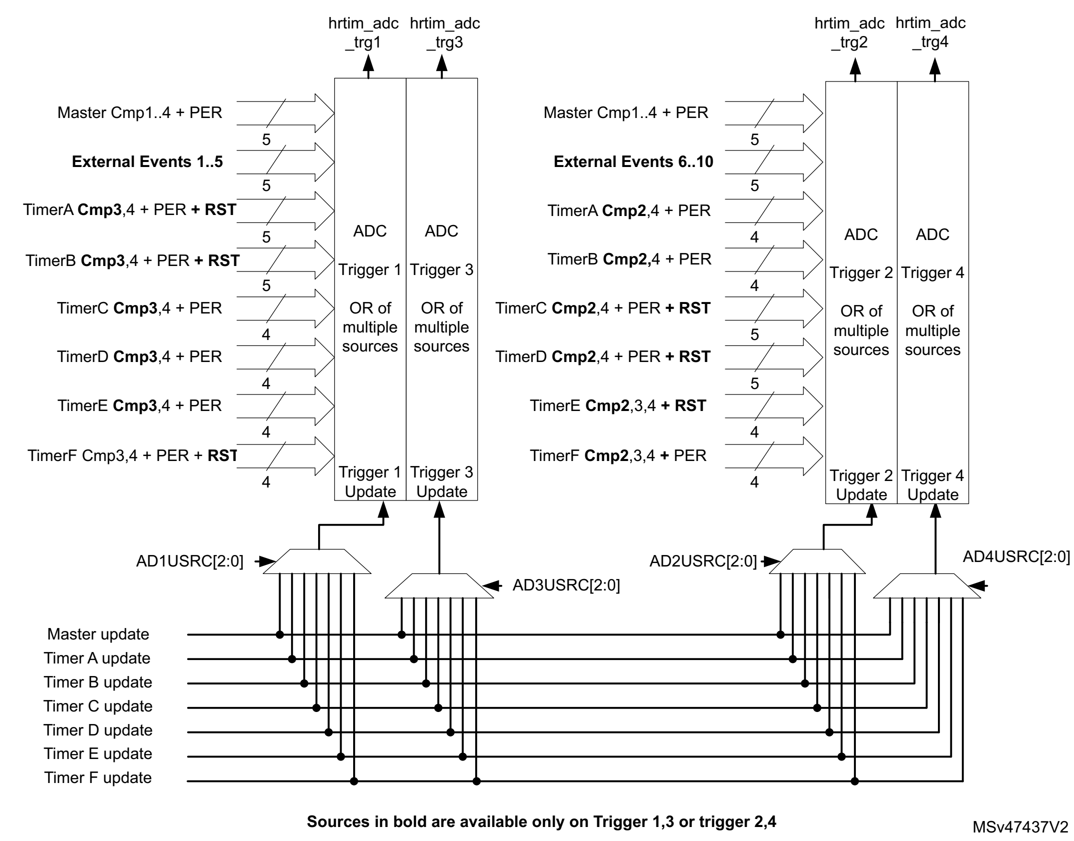
27.3.20 ADC triggers

The ADCs can be triggered by the master and the 6 timing units.

10 independent triggers are available for both the regular and the injected sequencers of the ADCs. The external events can be used as triggers. They are taken right after the conditioning defined in the HRTIM_EECRx registers, and are not depending on the EEFxR1 and EEFxR2 register settings.

Up to 32 events can be combined (ORed) for ADC triggers 1 to 4, in HRTIM_ADC1R to HRTIM_ADC4R registers, as shown on *Figure 251*. The ADC triggers 1/3 and 2/4 are using the same source set. A multiple triggering is possible within a single switching period by selecting several sources simultaneously. A typical use case is for a non-overlapping multiphase converter, where all phases can be sampled in a row using a single ADC trigger output.

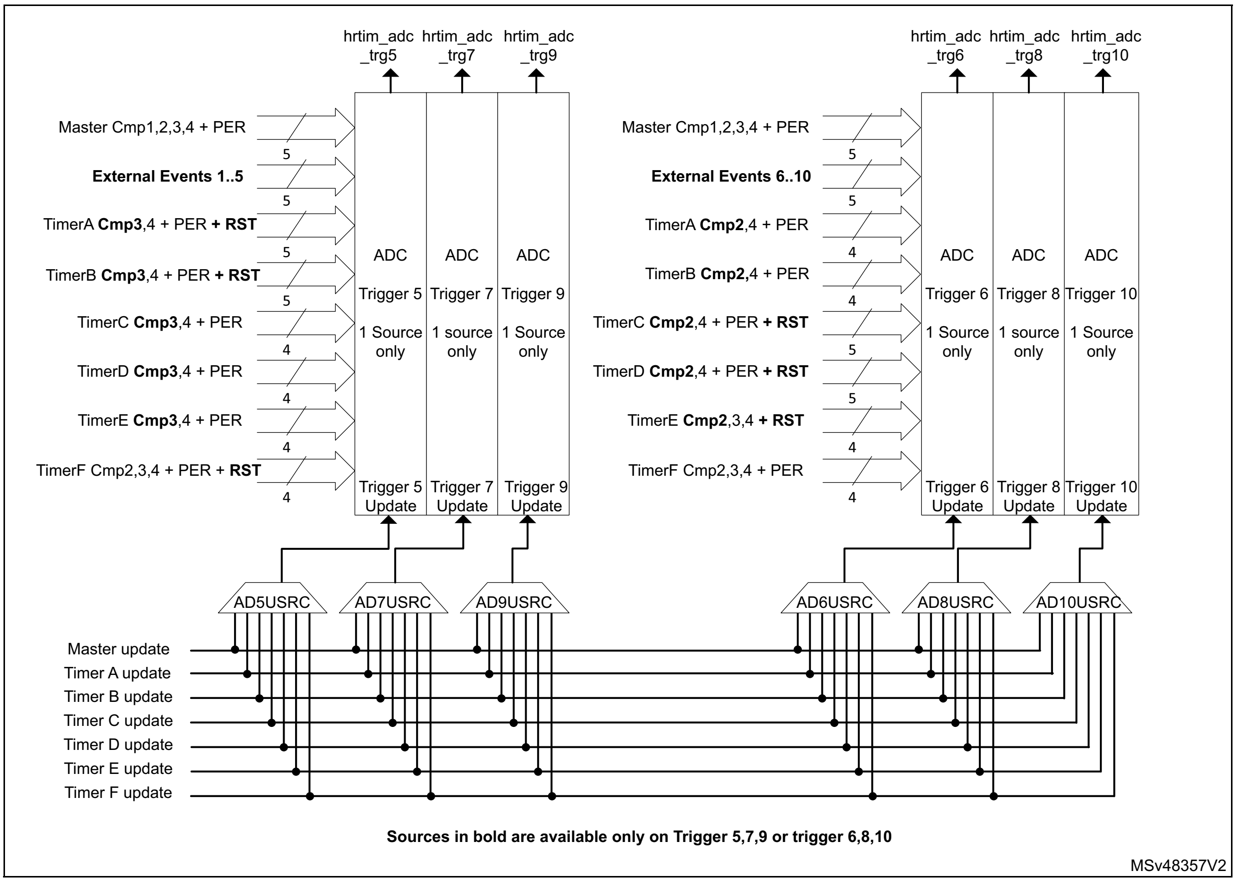
Figure 251. ADC trigger selection overview



The ADC triggers 5 to 10 are configured in the HRTIM_ADCER register, as shown on *Figure 252*. The ADC triggers 5/7/9 and 6/8/10 are using the same source set.

A single source can be selected at once for these triggers (1 out of 32 possible events).

Figure 252. ADC triggers



HRTIM_ADC1R to HRTIM_ADC4R and HRTIM_ADCER registers are preloaded and can be updated synchronously with the timer they are related to. The update sources are defined with ADxUSRC[2:0] bits in the HRTIM_CR1 and HRTIM_ADCUR registers.

For instance, if ADC trigger 1 outputs timer A CMP2 events (HRTIM_ADC1R = 0x0000 0400), HRTIM_ADC1R is typically updated simultaneously with timer A (AD1USRC[2:0] = 001).

When the preload is disabled (PREEN bit reset) in the source timer, the HRTIM_ADCxR registers are not preloaded either: a write access results in an immediate update of the trigger source.

ADC post-scaler

A post-scaling unit allows to reduce the ADC trigger rate as shown in *Figure 253* below.

Each ADC trigger rate can be individually adjusted using the ADCxPSC[4:0] bits in the HRTIM_ADCxPS1 and HRTIM_ADCxPS2 registers.

In the center-aligned mode, the ADC trigger rate is also dependent on ADROM[1:0] bitfield, programmed in the source timer, as shown in *Figure 254*. The ADROM[1:0] bitfield is coding for any event that can trigger the ADC: reset, roll-over (period) and compare event:

- ADROM[1:0] = 00: event generated both during up and down-counting phases
- ADROM[1:0] = 01: event generated during down-counting phases
- ADROM[1:0] = 10: event generated during up-counting phases

The ADC post-scaler programming register are preloaded and can be updated on-the-fly without stopping the timers.

Figure 253. ADC trigger post-scaling in up-counting mode

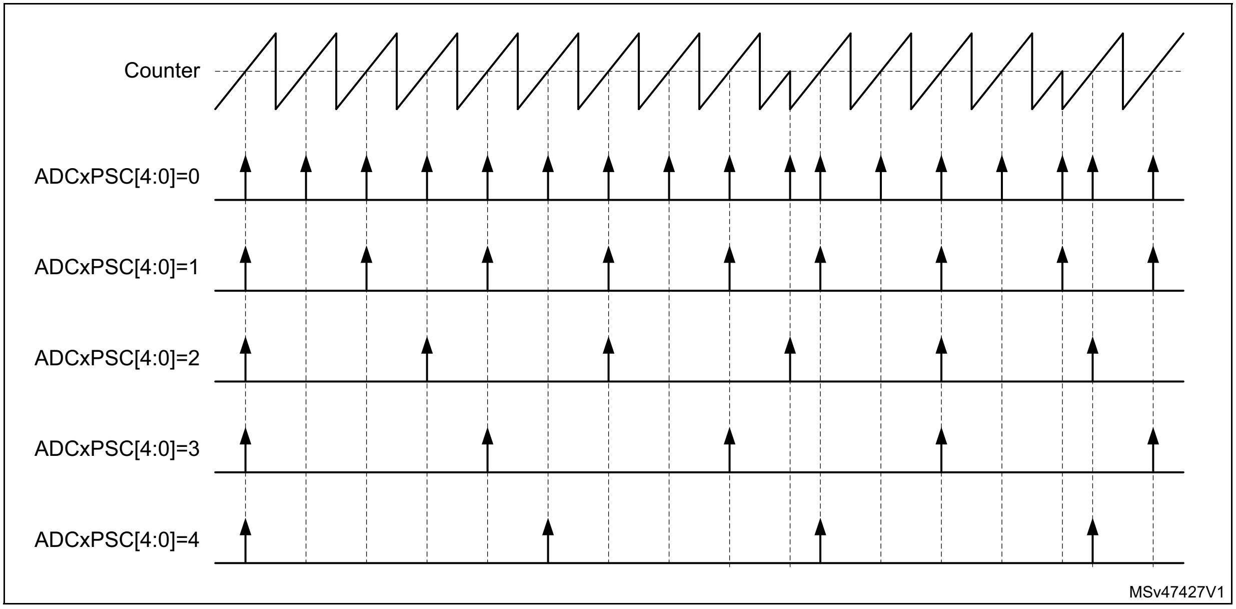
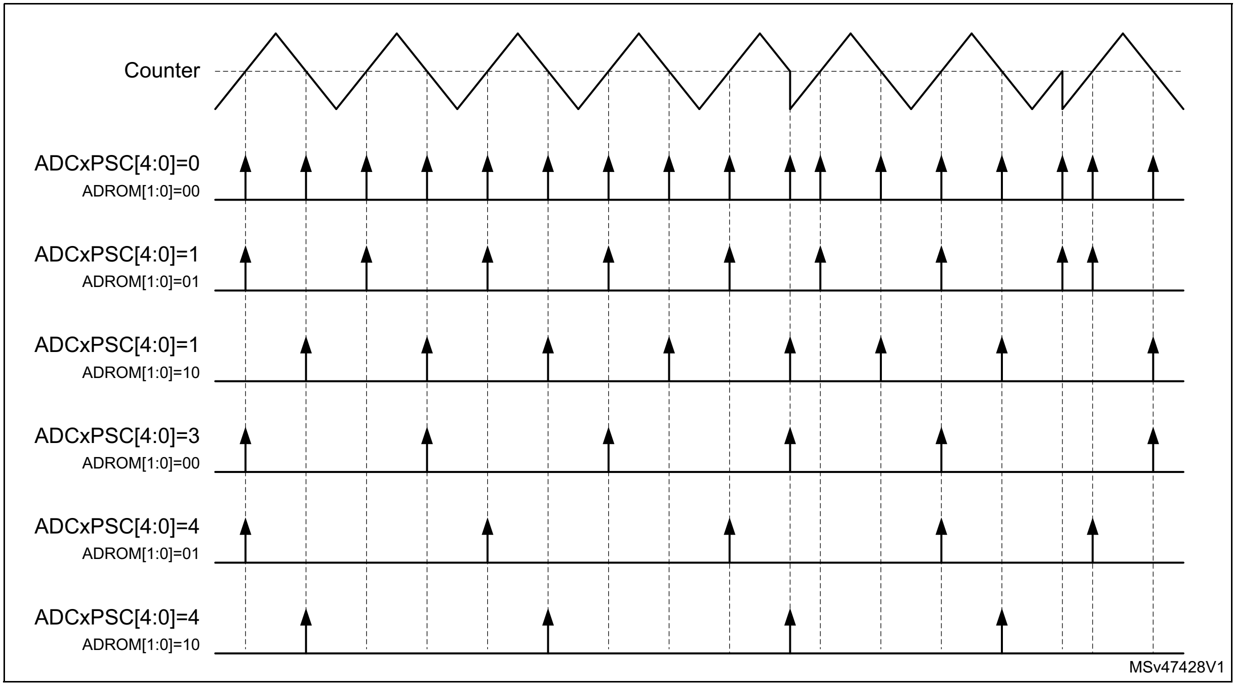


Figure 254. ADC trigger post-scaling in up/down counting mode



27.3.21 DAC triggers

The HRTIM allows to have the embedded DACs updated synchronously with the timer updates.

The update events from the master timer and the timer units can generate DAC update triggers on any of the 3 hrtim_dac_trgx outputs.

Note: Each timer has its own DAC-related control register.

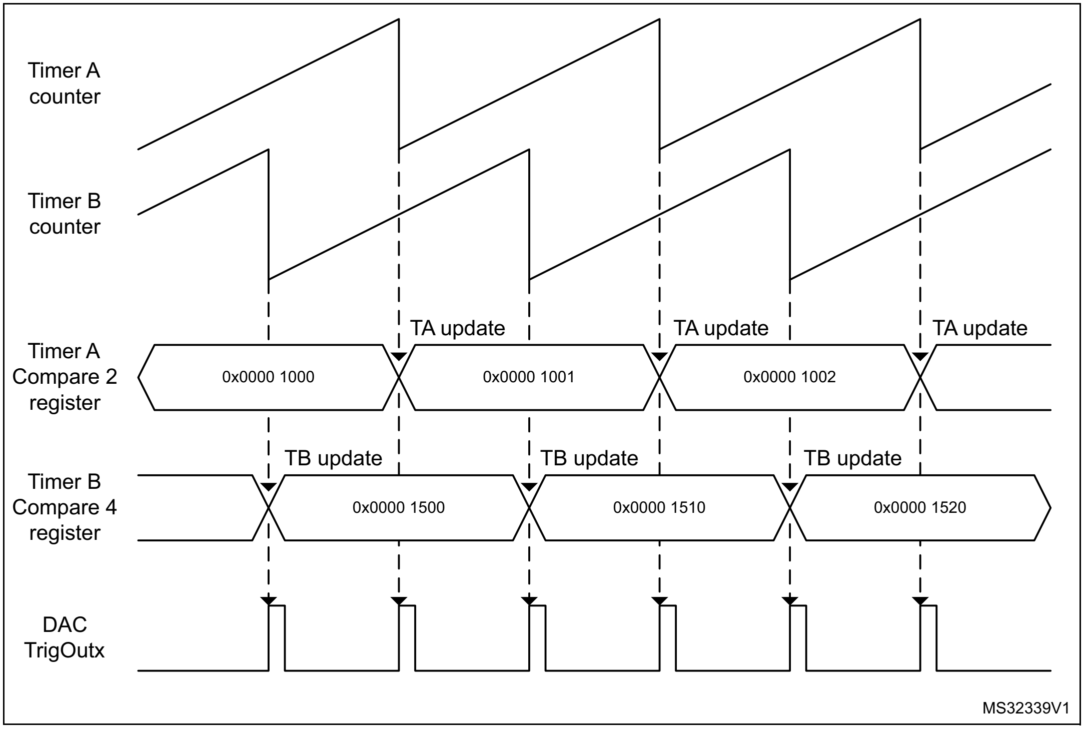
DACSYNC[1:0] bits of the HRTIM_MCR and HRTIM_TIMxCR registers are programmed as follows:

- 00: No update generated
- 01: Update generated on hrtim_dac_trg1
- 10: Update generated on hrtim_dac_trg2
- 11: Update generated on hrtim_dac_trg3

An output pulse of 1 f_{HRTIM} clock periods is generated on the hrtim_dac_trgx output.

When DACSYNC[1:0] bits are enabled in multiple timers, the hrtim_dac_trgx output consists of an OR of all timers' update events. For instance, if DACSYNC = 1 in timer A and in timer B, the update event in timer A is ORed with the update event in timer B to generate a DAC update trigger on the corresponding hrtim_dac_trgx output, as shown on *Figure 255*.

Figure 255. Combining several updates on a single hrtim_dac_trgx output



Refer to *Table 212: HRTIM DAC triggers connections* for connections to the DACs.

Dual channel DAC trigger

Slope compensation techniques and hysteretic control to be easily implemented using HRTIM built-in features and the DAC sawtooth generator. The principle is to have a DAC generating a decreasing saw-tooth synchronized with the PWM period, or a square wave synchronized with PWM signal.

This mode is enabled with the DCDE bit in the TIMxCR2 register. This bit cannot be changed once the timer is operating (TxEN bit set).

It uses two trigger outputs, as shown on the *Figure 256* below:

- the hrtim_dac_reset_trgx generates DAC reset/update events

- the hrtim_dac_step_trgx generates requests for incremental DAC value changes

The DCDR bit in the TIMxCR2 register defines when the hrtim_dac_reset_trgx trigger is generated:

- DCDR = 0: the trigger is generated on counter reset or roll-over event
- DCDR = 1: the trigger is generated on output 1 set event

Note: *The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).*

The DCDS bit in the TIMxCR2 register defines when the hrtim_dac_step_trgx trigger is generated:

- DCDS = 0: the trigger is generated on compare 2 event
- DCDS = 1: the trigger is generated on output 1 reset event

The DCDR and DCDS bits allows the following use cases to be covered:

- Edge-aligned slope compensation (DCDR = DCDS = 0): the DAC’s sawtooth starts on PWM period beginning and multiple triggers are generated during the period
- Center-aligned slope compensation (DCDR = 1 DCDS = 0): the DAC’s sawtooth starts on the output set event and multiple triggers are generated during the period
- Hysteretic controller: the DAC value must be changed twice per period, when the output state changes. 2 triggers are generated per PWM period. In edge-aligned mode (DCDR=0, DCDS =1), the triggers are generated on counter reset or roll-over. In center-aligned mode (DCDR=1, DCDS=1), the triggers are generated when the output is set.

The compare 2 has a particular operating mode when the DCDE is set and the DCDS bit is reset. The active comparison value is automatically updated as soon as a compare match has occured, so that the trigger can be repeated periodically with a period equal to the CMP2 value, as represented on *Figure 256*.
The dual channel DAC trigger with DCDS bit reset (compare 2 event used) must not be used simultaneously with modes using CMP2 (triple / quad interleaved and triggered-half modes).

Note: *The CMP2 value can be changed on-the-fly. The new value is taken into account on the next coming compare match.*

Note: *When the DCDS bit is reset, the CMP2 value must not be modified by other mechanisms: the interleaved, triggered half and balanced idle modes must be disabled.*

Table 238 below gives an example, for generating 6 triggers within a PWM period. It shows that it is necessary to round up the division result to the upper value.

Let’s consider a counter period TIMxPER = 8192. Dividing 8192 by 6 yields 1365.33.

– Round down value: 1365: 7 triggers are generated, the 6th and 7th being very close (respectively for counter = 8190 and 8192)

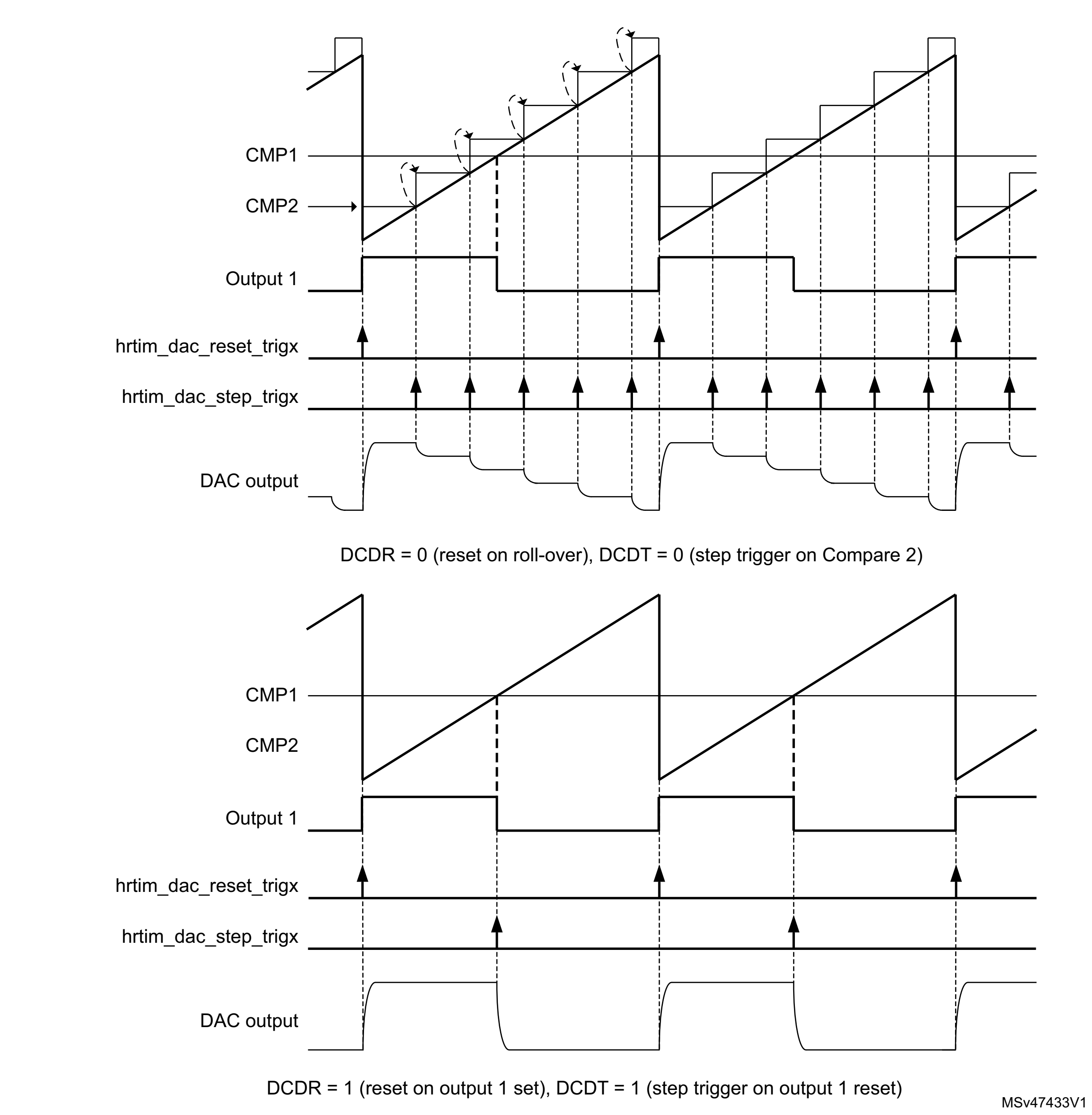
– Round up value:1366: 6 triggers are generated. The 6th trigger on dac_step_trg (for counter = 8192) is aborted by the counter roll-over from 8192 to 0.

Table 238. DAC dual channel trigger example

-	CMP2 = 1365	dac_trg	dac_step_trg	CMP2 = 1366	dac_trg	dac_step_trg
Counter value	1365	-	1	1366	-	1
	2730	-	2	2732	-	2
	4095	-	3	4098	-	3
	5460	-	4	5464	-	4
	6825	-	5	6830	-	5
	8190	-	6	8192	6	-
	8192	7	-	1366	-	1
	1365	-	1	2732	-	2
	...	-	-	...	-	-

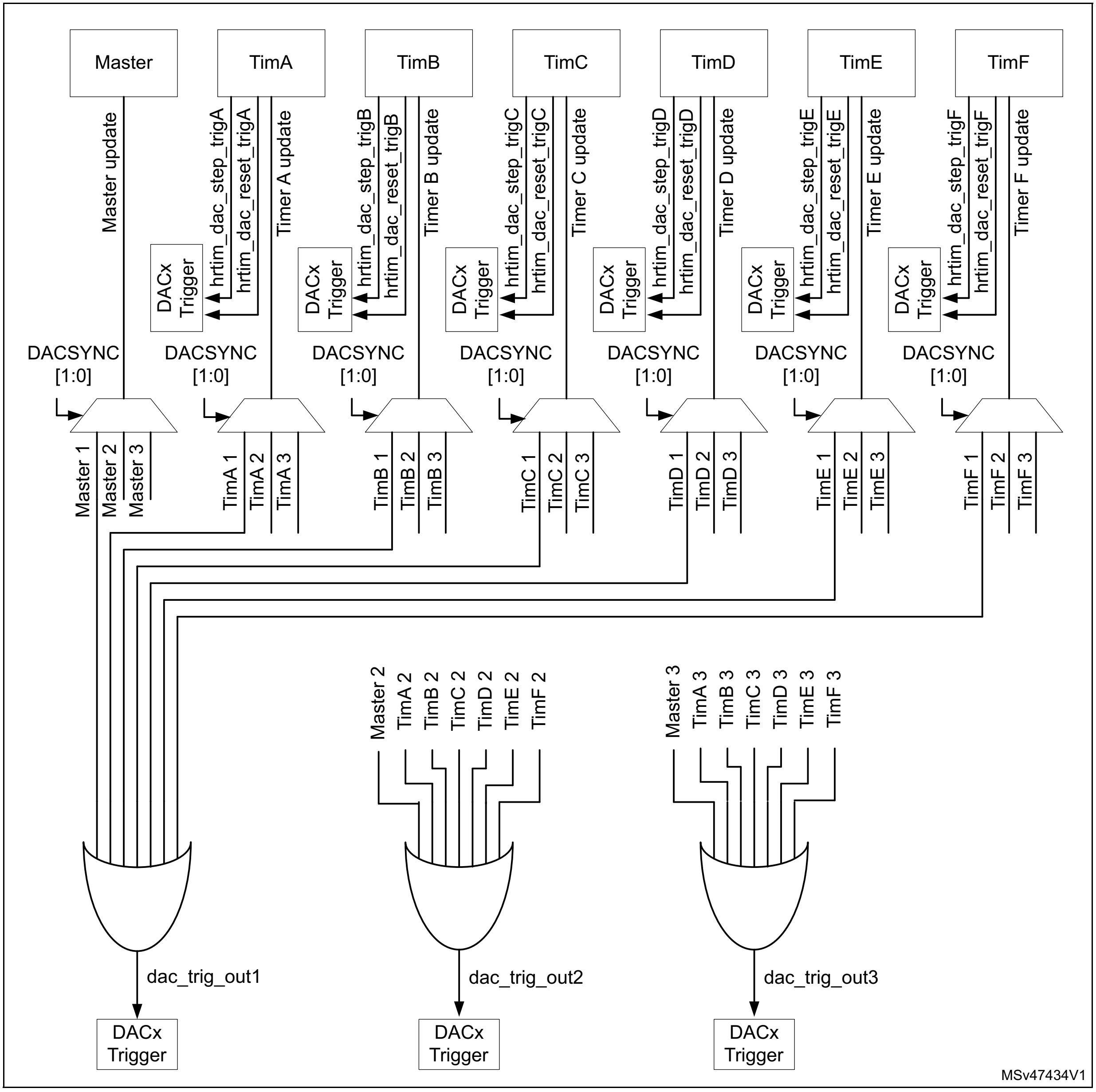
Note: In centered-pattern mode, it is mandatory to have an even number of triggers per switching period, so as to avoid unevenly spaced triggers around counter’s peak value.

Figure 256. DAC triggers for slope compensation



The *Figure 257* below provides an overview of all the available DAC triggers.

Figure 257. DAC triggers overview



27.3.22 Interrupts

7 interrupts can be generated by the master timer:

- Master timer registers update
- Synchronization event received
- Master timer repetition event
- Master compare 1 to 4 event

14 interrupts can be generated by each timing unit:

- Delayed protection triggered
- Counter reset or roll-over event
- Output 1 and output 2 reset (transition active to inactive)
- Output 1 and output 2 set (transition inactive to active)
- Capture 1 and 2 events
- Timing unit registers update
- Repetition event
- Compare 1 to 4 event

8 global interrupts are generated for the whole HRTIM:

- System fault and fault 1 to 6 (regardless of the timing unit attribution)
- DLL calibration done
- Burst mode period completed

The interrupt requests are grouped in 8 vectors as follows:

- hrtim_it1: master timer interrupts (master update, sync Input, repetition, MCMP1..4) and global interrupt except faults (burst mode period and DLL ready interrupts)
- hrtim_it2: TIMA interrupts
- hrtim_it3: TIMB interrupts
- hrtim_it4: TIMC interrupts
- hrtim_it5: TIMD interrupts
- hrtim_it6: TIME interrupts
- hrtim_it7: TIMF interrupts
- hrtim_it8: Dedicated vector all fault interrupts to allow high-priority interrupt handling

Table 239 is a summary of the interrupt requests, their mapping and associated control, and status bits.

Table 239. HRTIM interrupt summary

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
hrtim_it1	Burst mode period completed	BMPER	BMPERIE	BMPERC
	DLL calibration done	DLLRDY	DLLRDYIE	DLLRDYC
	Master timer registers update	MUPD	MUPDIE	MUPDC
	Synchronization event received	SYNC	SYNCIE	SYNCC
	Master timer repetition event	MREP	MREPIE	MREPC
	Master compare 1 to 4 event	MCMP1	MCMP1IE	MCP1C
		MCMP2	MCMP2IE	MCP2C
		MCMP3	MCMP3IE	MCP3C
		MCMP4	MCMP4IE	MCP4C

Table 239. HRTIM interrupt summary (continued)

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
hrtim_it2 hrtim_it3 hrtim_it4 hrtim_it5 hrtim_it6 hrtim_it7	Delayed protection triggered	DLYPRT	DLYPRTIE	DLYPRTC
	Counter reset or roll-over event	RST	RSTIE	RSTC
	Output 1 and output 2 reset (transition active to inactive)	RSTx1	RSTx1IE	RSTx1C
		RSTx2	RSTx2IE	RSTx2C
	Output 1 and output 2 set (transition inactive to active)	SETx1	SETx1IE	SETx1C
		SETx2	SETx2IE	SETx2C
	Capture 1 and 2 events	CPT1	CPT1IE	CPT1C
		CPT2	CPT2IE	CPT2C
	Timing unit registers update	UPD	UPDIE	UPDC
	Repetition event	REP	REPIE	REPC
	Compare 1 to 4 event	CMP1	CMP1IE	CMP1C
		CMP2	CMP2IE	CMP2C
		CMP3	CMP3IE	CMP3C
		CMP4	CMP4IE	CMP4C
hrtim_it8	System fault	SYSFLT	SYSFLTIE	SYSFLTC
	Fault 1 to 6	FLT1	FLT1IE	FLT1C
		FLT2	FLT2IE	FLT2C
		FLT3	FLT3IE	FLT3C
		FLT4	FLT4IE	FLT4C
		FLT5	FLT5IE	FLT5C
		FLT6	FLT6IE	FLT6C

27.3.23 DMA

Most of the events able to generate an interrupt can also generate a DMA request, even both simultaneously. Each timer (master, TIMA...F) has its own DMA enable register.

The individual DMA requests are ORed into 7 channels as follows:

- 1 channel for the master timer
- 1 channel per timing unit (TIMA...F)

Note: Before disabling a DMA channel (DMA enable bit reset in TIMxDIER), it is necessary to disable first the DMA controller.

Table 240 is a summary of the events with their associated DMA enable bits.

Table 240. HRTIM DMA request summary

DMA Channel	Event	DMA capable	DMA enable bit
hrtim_dma1 (master timer)	Burst mode period completed	No	N/A
	DLL calibration done	No	N/A
	Master timer registers update	Yes	MUPDDE
	Synchronization event received	Yes	SYNCDE
	Master timer repetition event	Yes	MREPDE
	Master compare 1 to 4 event	Yes	MCMP1DE
		Yes	MCMP2DE
		Yes	MCMP3DE
		Yes	MCMP4DE
hrtim_dma2 (timer A) hrtim_dma3 (timer B) hrtim_dma4 (timer C) hrtim_dma5 (timer D) hrtim_dma6 (timer E) hrtim_dma7 (timer F)	Delayed protection triggered	Yes	DLYPRTDE
	Counter reset or roll-over event	Yes	RSTDE
	Output 1 and output 2 reset (transition active to inactive)	Yes	RSTx1DE
		Yes	RSTx2DE
	Output 1 and output 2 set (transition inactive to active)	Yes	SETx1DE
		Yes	SETx2DE
	Capture 1 and 2 events	Yes	CPT1DE
		Yes	CPT2DE
	Timing unit registers update	Yes	UPDDE
	Repetition event	Yes	REPDE
	Compare 1 to 4 event	Yes	CMP1DE
		Yes	CMP2DE
		Yes	CMP3DE
		Yes	CMP4DE
N/A	System fault	No	N/A
	Fault 1 to 6	No	N/A
	Burst mode period completed	No	N/A
	DLL calibration done	No	N/A

Burst DMA transfers

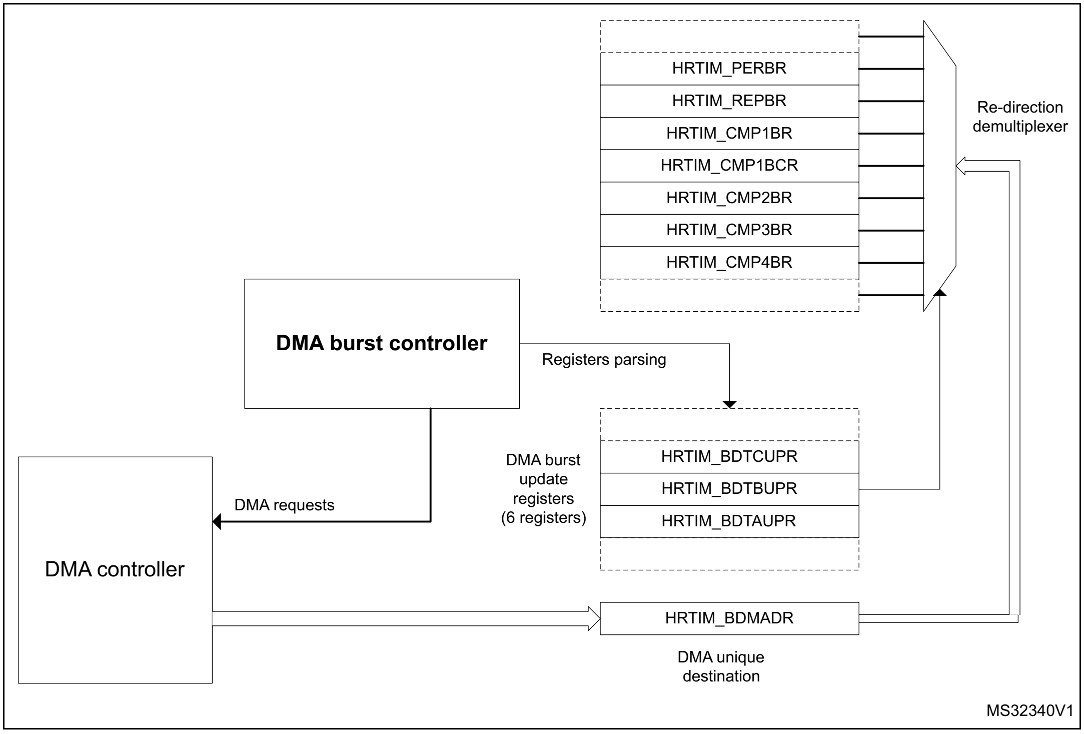
In addition to the standard DMA requests, the HRTIM features a DMA burst controller to have multiple registers updated with a single DMA request. This allows to:

- update multiple data registers with one DMA channel only,
- reprogram dynamically one or several timing units, for converters using multiple timer outputs.

The burst DMA feature is only available for one DMA channel, but any of the 6 channels can be selected for burst DMA transfers.

The principle is to program which registers are to be written by DMA. The master timer and TIMA..E have the burst DMA update register, where most of their control and data registers are associated with a selection bit: HRTIM_BDMUPR, HRTIM_BDTAUPR to HRTIM_BDTEUPR (this is applicable only for registers with write accesses). A redirection mechanism allows to forward the DMA write accesses to the HRTIM registers automatically, as shown on *Figure 258*.

Figure 258. DMA burst overview



When the DMA trigger occurs, the HRTIM generates multiple 32-bit DMA requests and parses the update register. If the control bit is set, the write access is redirected to the associated register. If the bit is reset, the register update is skipped and the register parsing is resumed until a new bit set is detected, to trigger a new request. Once the 6 update registers (HRTIM_BDMUPR, 5x HRTIM_BDTxUPR) are parsed, the burst is completed and the system is ready for another DMA trigger (see the flowchart on *Figure 259*).

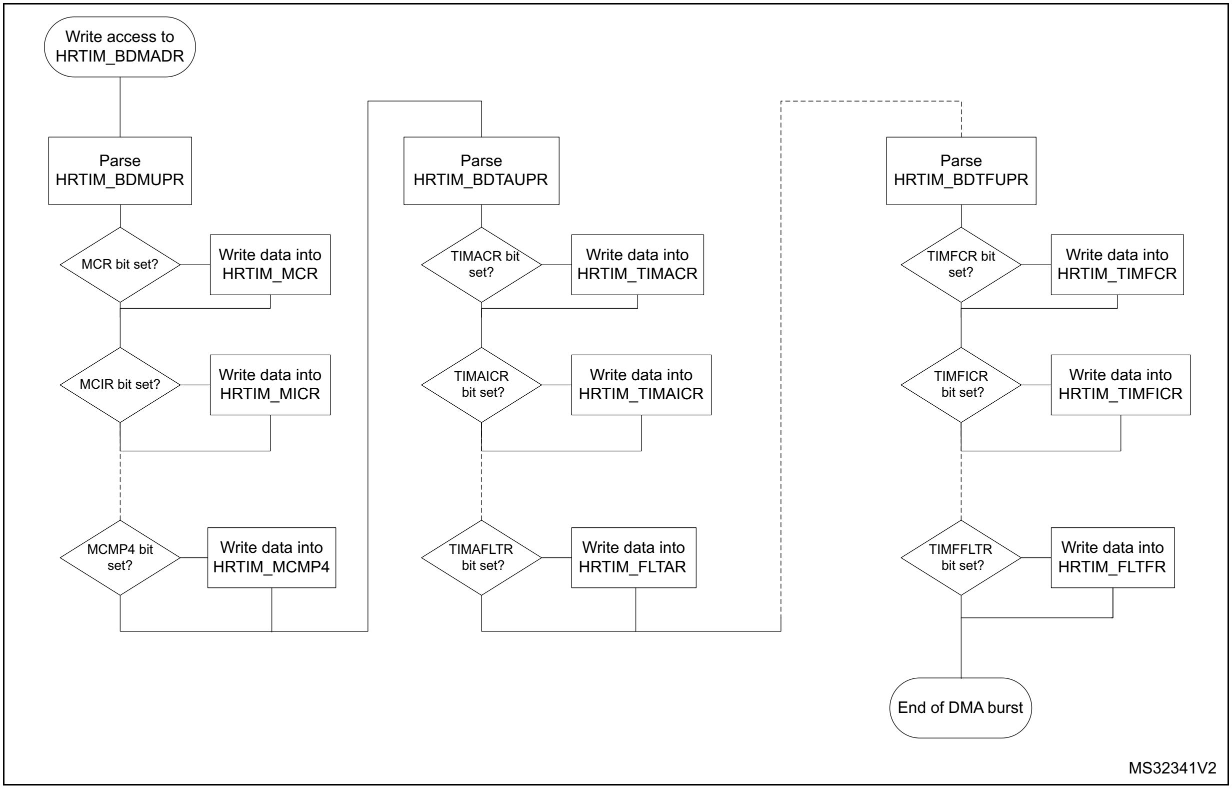
Note: Any trigger occurring while the burst is on-going is discarded, except if it occurs during the very last data transfer.

The burst DMA mode is permanently enabled (there is no enable bit). A burst DMA operation is started by the first write access into the HRTIM_BDMADR register.

It is only necessary to have the DMA controller pointing to the HRTIM_BDMADR register as the destination, in the memory, to the peripheral configuration with the peripheral increment mode disabled (the HRTIM handles internally the data re-routing to the final destination register).

To re-initialize the burst DMA mode if it was interrupted during a transaction, it is necessary to write at least to one of the 6 update registers.

Figure 259. Burst DMA operation flowchart


Write access to HRTIM_BDMADR
Parse HRTIM_BDMUPR
MCR bit set?
Write data into HRTIM_MCR
MCIR bit set?
Write data into HRTIM_MICR
MCMP4 bit set?
Write data into HRTIM_MCMP4
Parse HRTIM_BDTAUPR
TIMACR bit set?
Write data into HRTIM_TIMACR
TIMAICR bit set?
Write data into HRTIM_TIMAICR
TIMAFLTR bit set?
Write data into HRTIM_FLTAR
Parse HRTIM_BDTFUPR
TIMFCR bit set?
Write data into HRTIM_TIMFCR
TIMFICR bit set?
Write data into HRTIM_TIMFICR
TIMFFLTR bit set?
Write data into HRTIM_FLTFR
End of DMA burst
MS32341V2

Several options are available once the DMA burst is completed, depending on the register update strategy.

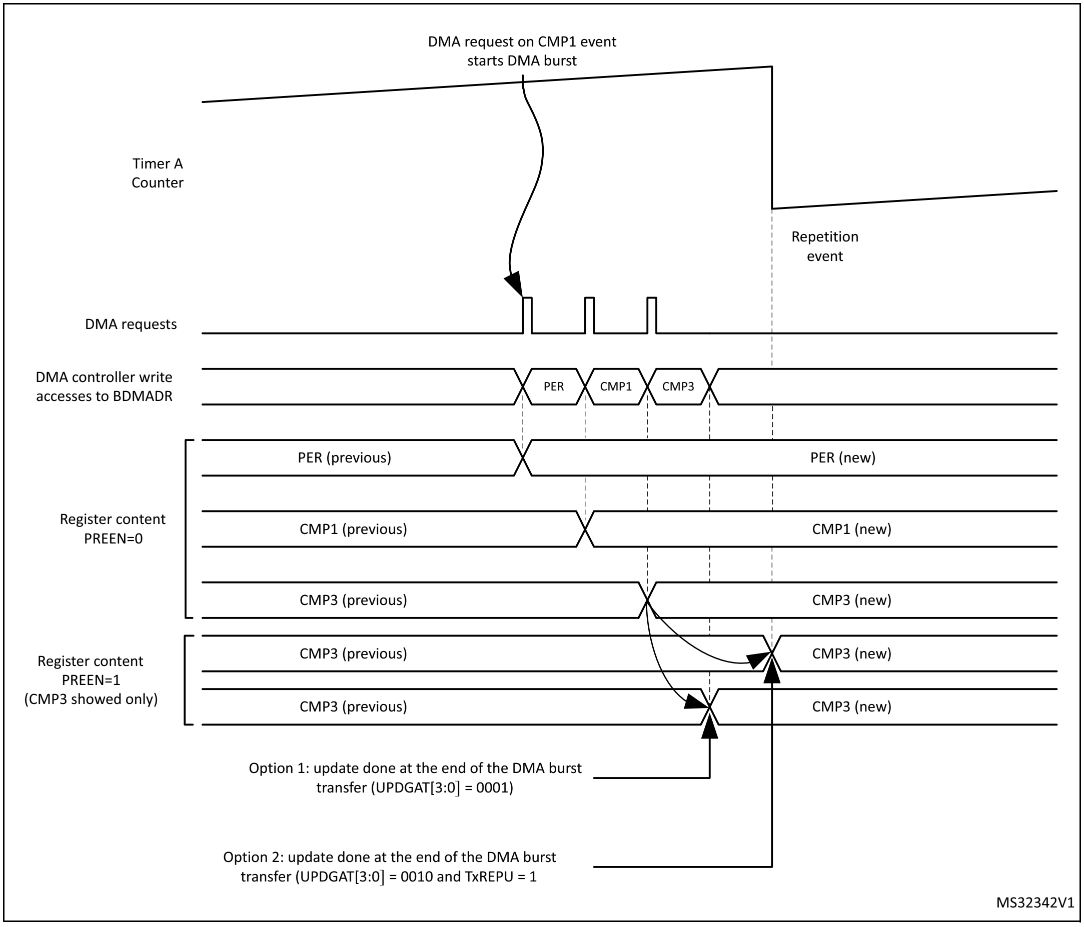
If the PREEN bit is reset (preload disabled), the value written by the DMA is immediately transferred into the active register and the registers are updated sequentially, following the DMA transaction pace.

When the preload is enabled (PREEN bit set), there are 3 use cases:

1. The update is done independently from DMA burst transfers (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 00 in HRTIM_MCR). In this case, and if it is necessary to have all transferred data taken into account simultaneously, the user must check that the DMA burst is completed before the update event takes place. On the contrary, if the update event happens while the DMA transfer is on-going, only part of the registers is loaded and the complete register update requires 2 consecutive update events.
2. The update is done when the DMA burst transfer is completed (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 01 in HRTIM_MCR). This mode guarantees that all new register values are transferred simultaneously. This is done independently from the counter value and can be combined with regular update events, if necessary (for instance, an update on a counter reset when TxRSTU is set).
3. The update is done on the update event following the DMA burst transfer completion (UPDGAT[3:0] = 0010 in HRTIM_TIMxCR and BRSTDMA[1:0] = 10 in HRTIM_MCR). This mode guarantees both a coherent update of all transferred data and the synchronization with regular update events, with the timer counter. In this case, if a regular update request occurs while the transfer is on-going, it is discarded and the effective update happens on the next coming update request.

The chronogram on *Figure 260* presents the active register content for 3 cases: PREEN=0, UPDGAT[3:0] = 0001 and UPDGAT[3:0] = 0001 (when PREEN = 1).

Figure 260. Registers update following DMA burst transfer


DMA request on CMP1 event starts DMA burst
Timer A Counter
Repetition event
DMA requests
DMA controller write accesses to BDMADR
PER
CMP1
CMP3
Register content PREEN=0
PER (previous)
PER (new)
CMP1 (previous)
CMP1 (new)
CMP3 (previous)
CMP3 (new)
Register content PREEN=1 (CMP3 showed only)
CMP3 (previous)
CMP3 (new)
CMP3 (previous)
CMP3 (new)
Option 1: update done at the end of the DMA burst transfer (UPDGAT[3:0] = 0001)
Option 2: update done at the end of the DMA burst transfer (UPDGAT[3:0] = 0010 and TxREPU = 1
MS32342V1

27.3.24 HRTIM initialization

This section describes the recommended HRTIM initialization procedure, including other related MCU peripherals.

The HRTIM clock source must be enabled in the reset and clock control unit (RCC), while respecting the $_{\text{fHRTIM}}$ range for the DLL lock.

The DLL calibration must be started by setting CAL bit in HRTIM_DLLCR register.

The HRTIM master and timing units can be started only once the high-resolution unit is ready. This is indicated by the DLLRDY flag set. The DLLRDY flag can be polled before resuming the initialization or the calibration can run in background while other registers of the HRTIM or other MCU peripherals are initialized. In this case, the DLLRDY flag must be checked before starting the counters (an end-of-calibration interrupt can be issued if necessary, enabled with DLLRDYIE flag in HRTIM_IER). Once the DLL calibration is done, CALEN bit must be set to have it done periodically and compensate for potential voltage and temperature drifts. The calibration periodicity is defined using the CALRTE[1:0] bitfield in the HRTIM_DLLCR register.

The HRTIM control registers can be initialized as per the power converter topology and the timing units use case. All inputs have to be configured (source, polarity, edge-sensitivity).

The HRTIM outputs must be set up eventually, with the following sequence:

- the polarity must be defined using POLx bits in HRTIM_OUTxR
- the FAULT and IDLE states must be configured using FAULTx[1:0] and IDLESx bits in HRTIM_OUTxR

The HRTIM outputs are ready to be connected to the MCU I/Os. In the GPIO controller, the selected HRTIM I/Os have to be configured as per the alternate function mapping table in the product datasheet.

From this point on, the HRTIM controls the outputs, which are in the IDLE state.

The outputs are configured in RUN mode by setting TxyOEN bits in the HRTIM_OENR register. The 2 outputs are in the inactive state until the first valid set/reset event in RUN mode. Any output set/reset event (except software requests using SST, SRT) are ignored as long as TxCEN bit is reset, as well as burst mode requests (IDLEM bit value is ignored). Similarly, any counter reset request coming from the burst mode controller is ignored (if TxBM bit is set).

Note: When the deadtime insertion is enabled (DTEN bit set), it is necessary to force the output state by software, using SST and RST bits, to have the outputs in a complementary state as soon as the RUN mode is entered.

The HRTIM operation can eventually be started by setting TxCEN or MCEN bits in HRTIM_MCR.

If the HRTIM peripheral is reset with the Reset and Clock Controller, the output control is released to the GPIO controller and the outputs are tri stated.

27.3.25 Debug

When a microcontroller enters the debug mode (Cortex®-M4 with FPU core halted), the TIMx counter either continues to work normally or stops, depending on DBG_HRTIM_STOP configuration bit in DBG module:

- DBG_HRTIM_STOP = 0: no behavior change, the HRTIM continues to operate.
- DBG_HRTIM_STOP = 1: all HRTIM timers, including the master, are stopped. The outputs in RUN mode enter the FAULT state if FAULTx[1:0] = 01,10,11, or keep their current state if FAULTx[1:0] = 00. The outputs in idle state are maintained in this state. This is permanently maintained even if the MCU exits the halt mode. This allows to maintain a safe state during the execution stepping. The outputs can be enabled again by settings TxyOEN bit (requires the use of the debugger).

Timer behavior during MCU halt when DBG_HRTIM_STOP = 1

The set/reset crossbar, the dead-time and push-pull unit, the idle/balanced fault detection and all the logic driving the normal output in RUN mode are not affected by debug. The output keeps on toggling internally, so as to retrieve regular signals of the outputs when TxyOEN is set again (during or after the MCU halt). Associated triggers and filters are also following internal waveforms when the outputs are disabled.

FAULT inputs and events (any source) are enabled during the MCU halt.

Fault status bits can be set and TxyOEN bits reset during the MCU halt if a fault occurs at that time (TxyOEN and TxyODS are not affected by DBG_HRTIM_STOP bit state).

Synchronization, counter reset, start and reset-start events are discarded in debug mode, as well as capture events. This is to keep all related registers stable as long as the MCU is halted.

The counter stops counting when a breakpoint is reached. However, the counter enable signal is not reset; consequently no start event is emitted when exiting from debug. All counter reset and capture triggers are disabled, as well as external events (ignored as long as the MCU is halted). The outputs SET and RST flags are frozen, except in case of forced software set/reset. A level-sensitive event is masked during the debug but is active again as soon as the debug is exited. For edge-sensitive events, if the signal is maintained active during the MCU halt, a new edge is not generated when exiting from debug.

The update events are discarded. This prevents any update trigger on hrtim_upd_en[3:1] inputs. DMA triggers are disabled. The burst mode circuit is frozen: the triggers are ignored and the burst mode counter stopped.

DLL calibration is not blocked while the MCU is halted (the DLLRDY flag can be set).

27.4 Application use cases

27.4.1 Buck converter

Buck converters are of common use as step-down converters. The HRTIM can control up to 12 buck converters with 7 independent switching frequencies.

The converter usually operates at a fixed frequency and the Vin/Vout ratio depends on the duty cycle D applied to the power switch:.

$$V_{out} = D \times V_{in}$$

The topology is given on *Figure 261* with the connection to the ADC for voltage reading.

Figure 261. Buck converter topology

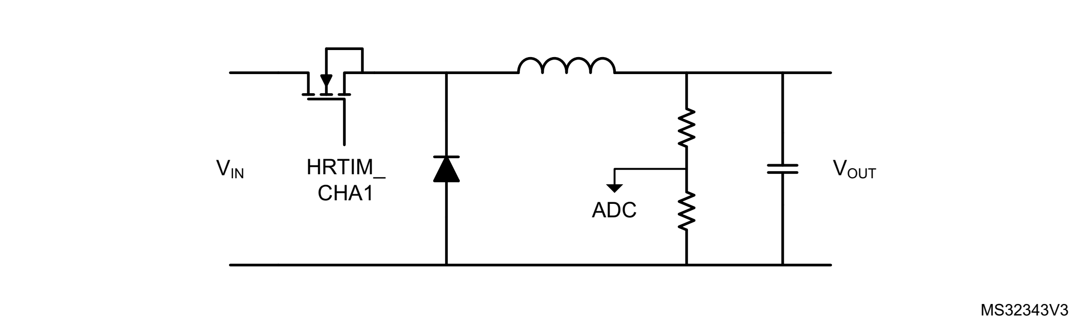
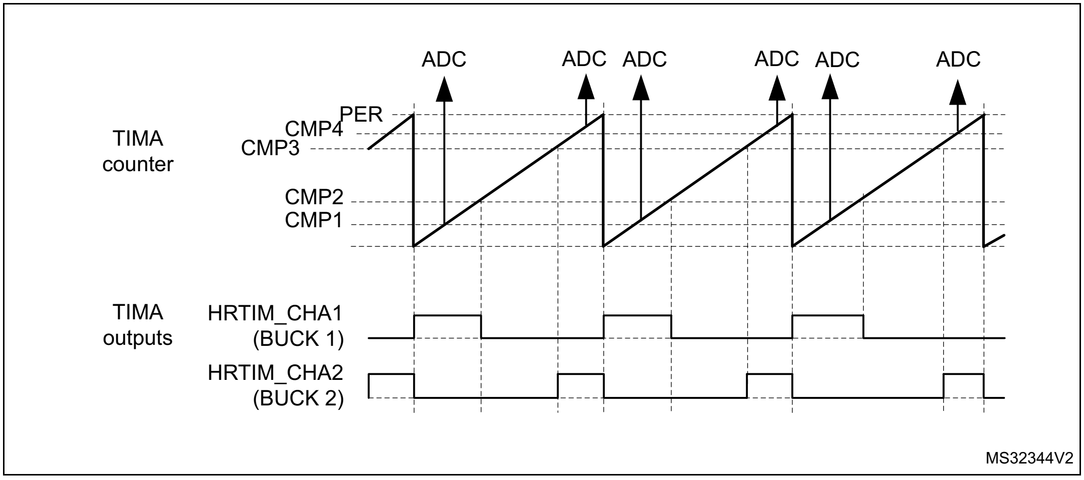


Figure 262 presents the management of two converters with identical frequency PWM signals. The outputs are defined as follows:

- HRTIM_CHA1 set on period, reset on CMP1
- HRTIM_CHA2 set on CMP3, reset on PER

The ADC is triggered twice per period, precisely in the middle of the ON time, using CMP2 and CMP4 events.

Figure 262. Dual Buck converter management

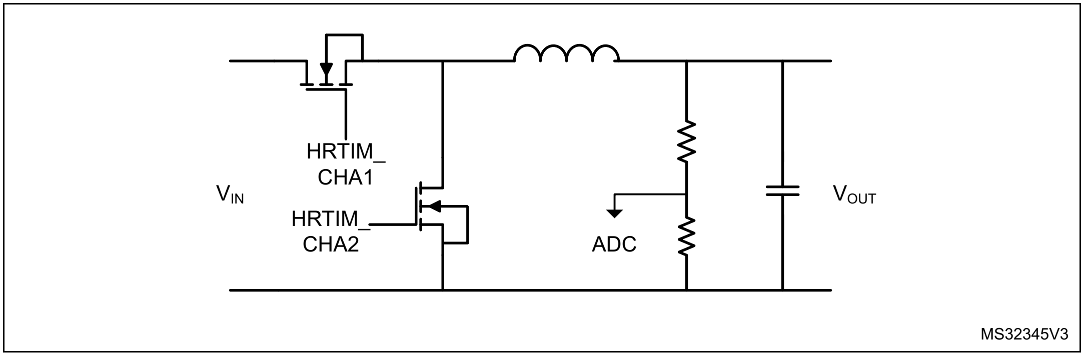


Timers A..E provide either 12 buck converters coupled by pairs (both with identical switching frequencies) or 7 completely independent converters (each of them having a different switching frequency), using the master timer as the 7th time base.

27.4.2 Buck converter with synchronous rectification

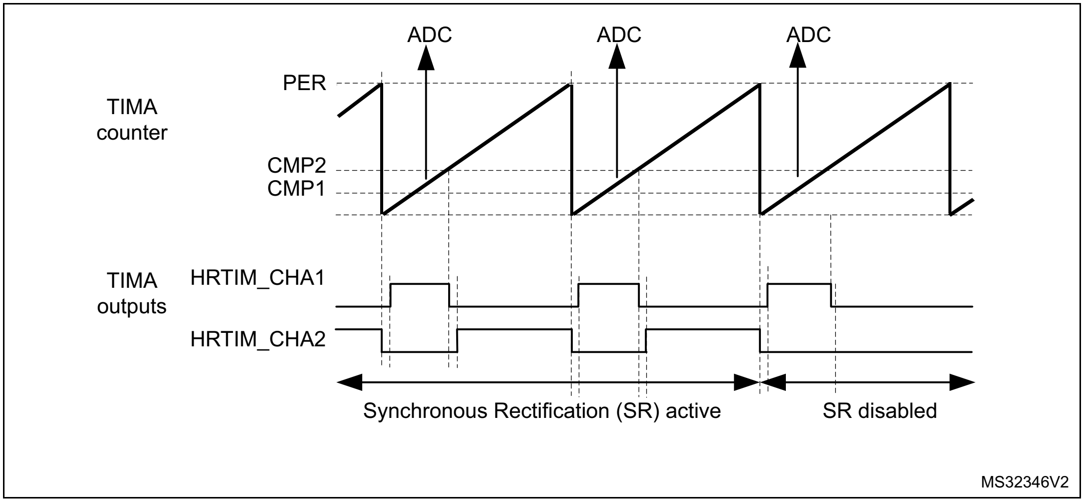
Synchronous rectification allows to minimize losses in buck converters, by means of a FET replacing the freewheeling diode. Synchronous rectification can be turned on or off on the fly depending on the output current level, as shown on *Figure 263*.

Figure 263. Synchronous rectification depending on output current



The main difference versus a single-switch buck converter is the addition of a deadtime for an almost complementary waveform generation on HRTIM_CHA2, based on the reference waveform on HRTIM_CHA1 (see *Figure 264*).

Figure 264. Buck with synchronous rectification



27.4.3 Multiphase converters

Multiphase techniques can be applied to multiple power conversion topologies (buck, flyback). Their main benefits are:

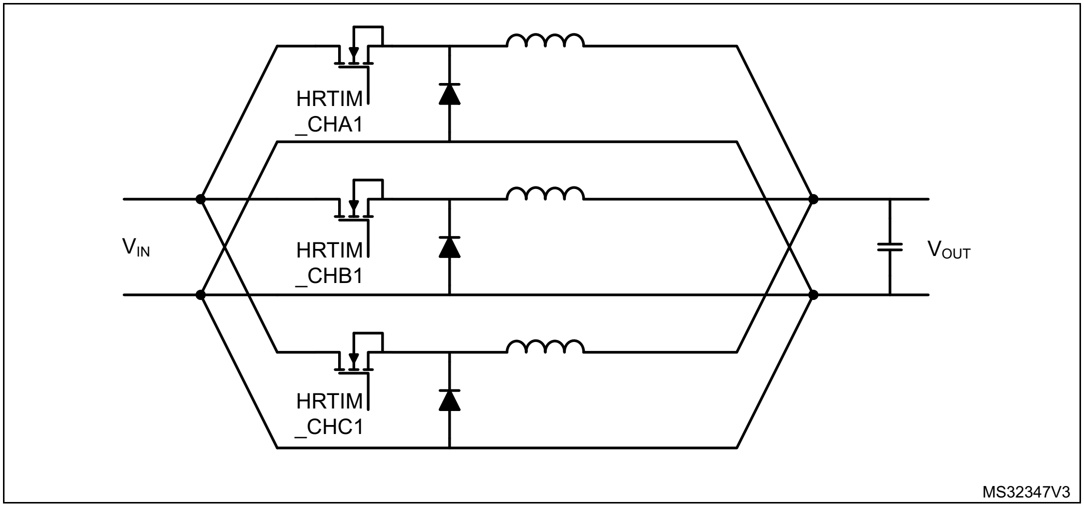
- Reduction of the current ripple on the input and output capacitors
- Reduced EMI
- Higher efficiency at light load by dynamically changing the number of phases (phase shedding)

The HRTIM manages multiple converters. The number of converters that can be controlled depends on the topologies and resources used (including the ADC triggers):

- 5 buck converters with synchronous rectification (SR), using the master timer and the 5 timers
- 4 buck converters (without SR), using the master timer and 2 timers

Figure 265 presents the topology of a 3-phase interleaved buck converter.

Figure 265. 3-phase interleaved buck converter



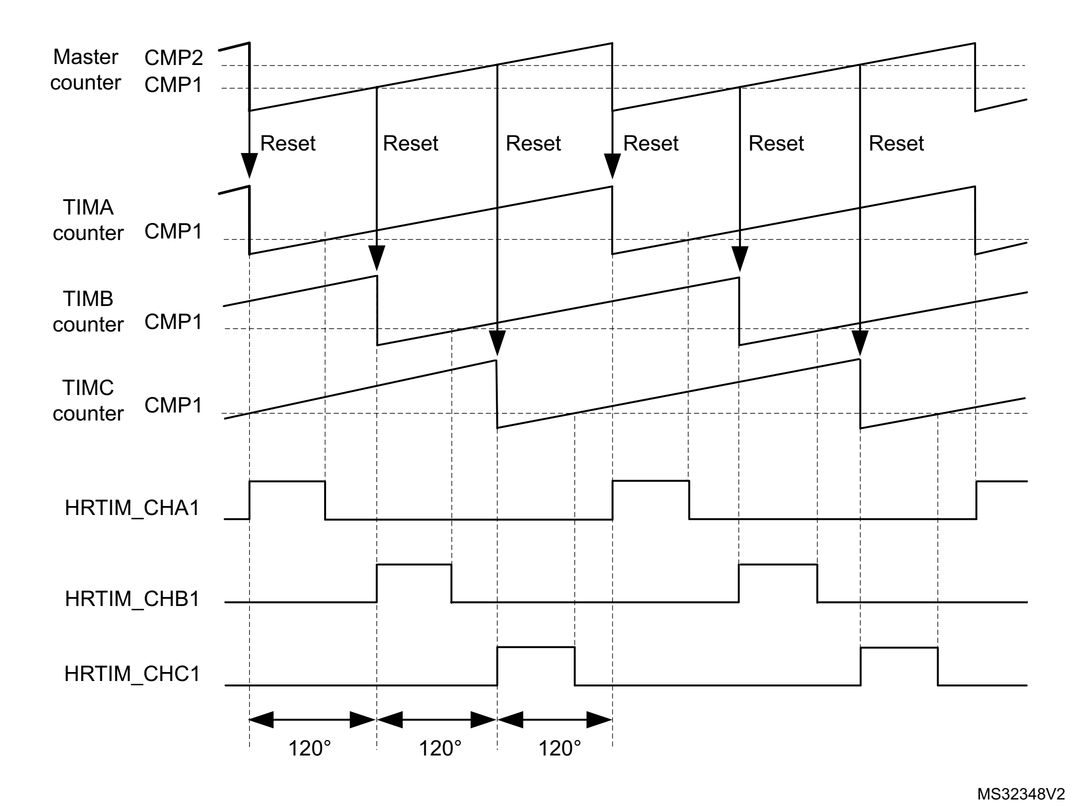
The master timer is responsible for the phase management: it defines the phase relationship between the converters by resetting the timers periodically. The phase-shift is 360° divided by the number of phases, 120° in the given example.

The duty cycle is then programmed into each of the timers. The outputs are defined as follows:

- HRTIM_CHA1 set on master timer period, reset on TACMP1
- HRTIM_CHB1 set on master timer MCMP1, reset on TBCMP1
- HRTIM_CHC1 set on master timer MCMP2, reset on TCCMP1

The ADC trigger can be generated on TxCMP2 compare event. Since all ADC trigger sources are phase-shifted because of the converter topology, it is possible to have all of them combined into a single ADC trigger to save ADC resources (for instance 1 ADC regular channel for the full multi-phase converter).

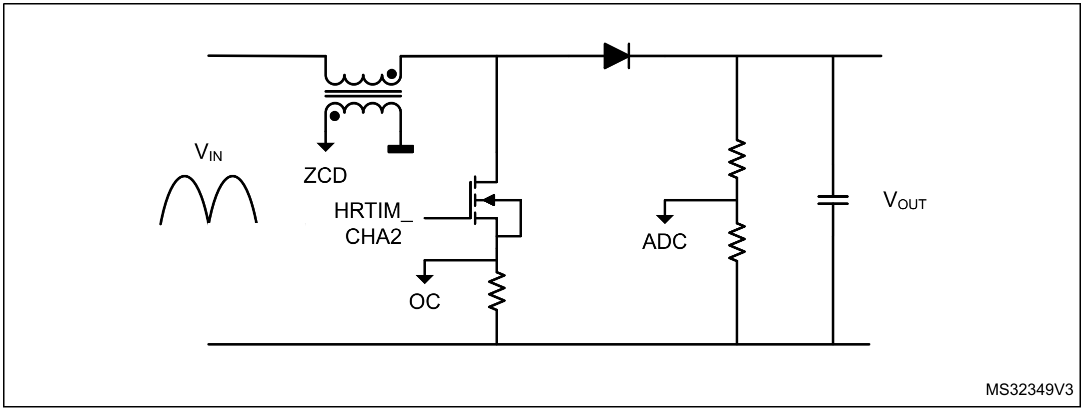
Figure 266. 3-phase interleaved buck converter control



27.4.4 Transition mode power factor correction

The basic operating principle is to build up current into an inductor during a fixed Ton time. This current then decays during the Toff time, and the period is re-started when it becomes null. This is detected using a Zero Crossing Detection circuitry (ZCD), as shown on *Figure 267*. With a constant Ton time, the peak current value in the inductor is proportional to the rectified AC input voltage, which provides the power factor correction.

Figure 267. Transition mode PFC



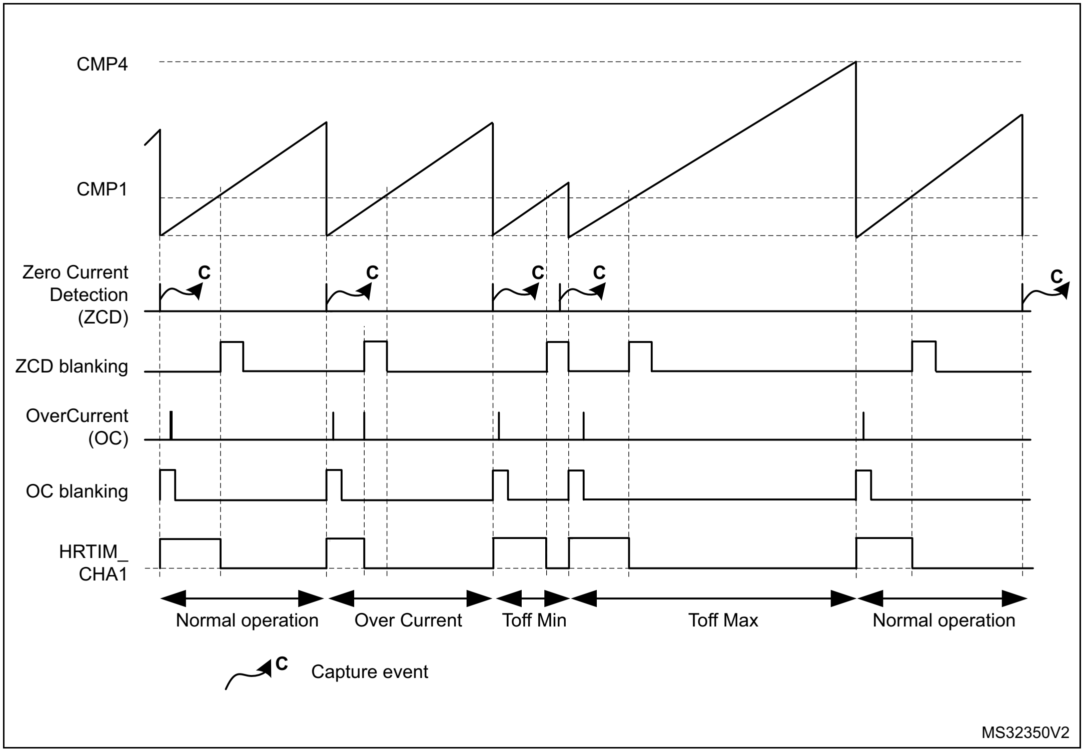
This converter operates with a constant Ton time and a variable frequency due the Toff time variation (depending on the input voltage). It must also include some features to operate when no zero-crossing is detected, or to limit the Ton time in case of over-current (OC). The OC feedback is usually conditioned with the built-in comparator and routed onto an external event input.

Figure 268 presents the waveform during the various operating modes, with the following defined parameters:

- Ton Min: masks spurious overcurrent (freewheeling diode recovery current), represented as OC blanking
- Ton Max: practically, the converter set-point. It is defined by CMP1
- Toff Min: limits the frequency when the current limit is close to zero (demagnetization is very fast). It is defined with CMP2.
- Toff Max: prevents the system to be stuck if no ZCD occurs. It is defined with CMP4 in auto-delayed mode.

Both Toff values are auto-delayed since the value must be relative to the output falling edge.

Figure 268. Transition mode PFC waveforms



27.5 HRTIM registers

27.5.1 HRTIM master timer control register (HRTIM_MCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	38	27	26	25	24	23	22	21	20	19	18	17	16
BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	TFCEN	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN
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
SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		INTLVD[1:0]		HALF	RETRIG	CONT	CKPSC[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **BRSTDMA[1:0]**: Burst DMA update

These bits define how the update occurs relatively to a burst DMA transaction.

00: Update done independently from the DMA burst transfer completion

01: Update done when the DMA burst transfer is completed

10: Update done on master timer roll-over following a DMA burst transfer completion. This mode only works in continuous mode.

11: Reserved

Bit 29 **MREPU**: Master timer repetition update

This bit defines whether an update occurs when the master timer repetition period is completed (either due to roll-over or reset events). MREPU can be set only if BRSTDMA[1:0] = 00 or 01.

0: Update on repetition disabled

1: Update on repetition enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **PREEN**: Preload enable

This bit enables the registers preload mechanism and defines whether the write accesses to the memory mapped registers are done into HRTIM's active or preload registers.

0: Preload disabled: the write access is directly done into the active register

1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]** DAC synchronization

A DAC synchronization event can be enabled and generated when the master timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 27.3.21: DAC triggers* for connections details).

00: No DAC trigger generated

01: Trigger generated on hrtim_dac_trg1

10: Trigger generated on hrtim_dac_trg2

11: Trigger generated on hrtim_dac_trg3

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **TFCEN**: Timer F counter enable

This bit starts the timer F counter.

0: Timer F counter disabled

1: Timer F counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 21 **TECEN**: Timer E counter enable

This bit starts the timer E counter.
0: Timer E counter disabled
1: Timer E counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 20 **TDCEN**: Timer D counter enable

This bit starts the timer D counter.
0: Timer D counter disabled
1: Timer D counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 19 **TCCEN**: Timer C counter enable

This bit starts the timer C counter.
0: Timer C counter disabled
1: Timer C counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 18 **TBCEN**: Timer B counter enable

This bit starts the timer B counter.
0: Timer B counter disabled
1: Timer B counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 17 **TACEN**: Timer A counter enable

This bit starts the timer A counter.
0: Timer A counter disabled
1: Timer A counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 16 **MCEN**: Master timer counter enable

This bit starts the master timer counter.
0: Master counter disabled
1: Master counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bits 15:14 **SYNCSRC[1:0]**: Synchronization source

These bits define the source and event to be sent on the synchronization outputs SYNCOUT[2:1]
00: Master timer start
01: Master timer compare 1 event
10: Timer A start/reset
11: Timer A compare 1 event

Bits 13:12 **SYNCOUT[1:0]**: Synchronization output

These bits define the routing and conditioning of the synchronization output event.
00: Disabled
01: Reserved.
10: Positive pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
11: Negative pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 11 **SYNCSTRTM**: Synchronization starts master

This bit enables the master timer start when receiving a synchronization input event:
0: No effect on the master timer
1: A synchronization input event starts the master timer

Bit 10 **SYNCRSTM**: Synchronization resets master

This bit enables the master timer reset when receiving a synchronization input event:

0: No effect on the master timer

1: A synchronization input event resets the master timer

Bits 9:8 **SYNCIN[1:0]** Synchronization input

These bits are defining the synchronization input source.

00: Disabled. HRTIM is not synchronized and runs in standalone mode.

01: Reserved.

10: Internal event on hrtim_in_sync[2]: the HRTIM is synchronized with the on-chip timer (see *Section : Synchronization input*).

11: External event on hrtim_in_sync[3]: a positive pulse on HRTIM_SCIN input triggers the HRTIM.

Note: This parameter cannot be changed once the impacted timers are enabled.

Bits 7:6 **INTLVD[1:0]**: Interleaved mode

This bitfield is significant only when the HALF bit is reset. It enables the interleaved mode.

00: Interleaved mode disabled

01: Triple interleaved mode: when HRTIM_MPER register is written, the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/3 value, and the HRTIM_MCMP2R active register is automatically updated with 2x (HRTIM_MPER/3) value.

10: Quad interleaved mode: when HRTIM_MPER register is written, the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/4 value, the HRTIM_MCMP2R active register is automatically updated with HRTIM_MPER/2 value and the HRTIM_MCMP3R active register is automatically updated with 3x (HRTIM_MPER/4) value.

11: Interleaved mode disabled

Bit 5 **HALF**: Half mode

This bit enables the half duty-cycle mode: the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/2 value when HRTIM_MPER register is written.

0: Half mode disabled

1: Half mode enabled

Bit 4 **RETRIG**: Re-triggerable mode

This bit defines the behavior of the master timer counter in single-shot mode.

0: The timer is not re-triggerable: a counter reset can be done only if the counter is stopped (period elapsed)

1: The timer is re-triggerable: a counter reset is done whatever the counter state (running or stopped)

Bit 3 **CONT**: Continuous mode

0: The timer operates in single-shot mode and stops when it reaches the MPER value

1: The timer operates in continuous (free-running) mode and rolls over to zero when it reaches the MPER value

Bits 2:0 **CKPSC[2:0]**: Clock prescaler

These bits define the master timer high-resolution clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to f_{HRCK} / $2^{CKPSC[2:0]}$.

The prescaling ratio cannot be modified once the timer is enabled.

27.5.2 HRTIM master timer interrupt status register (HRTIM_MISR)

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.	MUPD	SYNC	MREP	MCMP 4	MCMP 3	MCMP 2	MCMP 1
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPD**: Master update interrupt flag

This bit is set by hardware when the master timer registers are updated.
0: No master update interrupt occurred
1: Master update interrupt occurred

Bit 5 **SYNC**: Sync input interrupt flag

This bit is set by hardware when a synchronization input event is received.
0: No sync input interrupt occurred
1: Sync input interrupt occurred

Bit 4 **MREP**: Master repetition interrupt flag

This bit is set by hardware when the master timer repetition period has elapsed.
0: No master repetition interrupt occurred
1: Master repetition interrupt occurred

Bit 3 **MCMP4**: Master compare 4 interrupt flag

Refer to MCMP1 description

Bit 2 **MCMP3**: Master compare 3 interrupt flag

Refer to MCMP1 description

Bit 1 **MCMP2**: Master compare 2 interrupt flag

Refer to MCMP1 description

Bit 0 **MCMP1**: Master compare 1 interrupt flag

This bit is set by hardware when the master timer counter matches the value programmed in the master compare 1 register.
0: No master compare 1 interrupt occurred
1: Master compare 1 interrupt occurred

27.5.3 HRTIM master timer interrupt clear register (HRTIM_MICR)

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.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPDC**: Master update interrupt flag clear

Writing 1 to this bit clears the MUPDC flag in HRTIM_MISR register.

Bit 5 **SYNCC**: Sync input interrupt flag clear

Writing 1 to this bit clears the SYNC flag in HRTIM_MISR register.

Bit 4 **MREPC**: Repetition interrupt flag clear

Writing 1 to this bit clears the MREP flag in HRTIM_MISR register.

Bit 3 **MCMP4C**: Master compare 4 interrupt flag clear

Writing 1 to this bit clears the MCMP4 flag in HRTIM_MISR register.

Bit 2 **MCMP3C**: Master compare 3 interrupt flag clear

Writing 1 to this bit clears the MCMP3 flag in HRTIM_MISR register.

Bit 1 **MCMP2C**: Master compare 2 interrupt flag clear

Writing 1 to this bit clears the MCMP2 flag in HRTIM_MISR register.

Bit 0 **MCMP1C**: Master compare 1 interrupt flag clear

Writing 1 to this bit clears the MCMP1 flag in HRTIM_MISR register.

27.5.4 HRTIM master timer DMA interrupt enable register (HRTIM_MDIER)

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.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE
									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.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
									rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **MUPDDE**: Master update DMA request enable

This bit is set and cleared by software to enable/disable the master update DMA requests.

0: Master update DMA request disabled

1: Master update DMA request enabled

Bit 21 **SYNCDE**: Sync input DMA request enable

This bit is set and cleared by software to enable/disable the sync input DMA requests.

0: Sync input DMA request disabled

1: Sync input DMA request enabled

Bit 20 **MREPDE**: Master repetition DMA request enable

This bit is set and cleared by software to enable/disable the master timer repetition DMA requests.

0: Repetition DMA request disabled

1: Repetition DMA request enabled

Bit 19 **MCMP4DE**: Master compare 4 DMA request enable

Refer to MCMP1DE description

Bit 18 **MCMP3DE**: Master compare 3 DMA request enable

Refer to MCMP1DE description

Bit 17 **MCMP2DE**: Master compare 2 DMA request enable

Refer to MCMP1DE description

Bit 16 **MCMP1DE**: Master compare 1 DMA request enable

This bit is set and cleared by software to enable/disable the master timer compare 1 DMA requests.

0: Compare 1 DMA request disabled

1: Compare 1 DMA request enabled

Bits 15:6 Reserved, must be kept at reset value.

Bit 6 **MUPDIE**: Master update interrupt enable

This bit is set and cleared by software to enable/disable the master timer registers update interrupts

0: Master update interrupts disabled

1: Master update interrupts enabled

Bit 5 **SYNCIE**: Sync input interrupt enable

This bit is set and cleared by software to enable/disable the sync input interrupts

0: Sync input interrupts disabled

1: Sync input interrupts enabled

Bit 4 **MREPIE**: Master repetition interrupt enable

This bit is set and cleared by software to enable/disable the master timer repetition interrupts

0: Master repetition interrupt disabled

1: Master repetition interrupt enabled

Bit 3 **MCMP4IE**: Master compare 4 interrupt enable

Refer to MCMP1IE description

Bit 2 **MCMP3IE**: Master compare 3 interrupt enable

Refer to MCMP1IE description

Bit 1 **MCMP2IE**: Master compare 2 interrupt enable

Refer to MCMP1IE description

Bit 0 **MCMP1IE**: Master compare 1 interrupt enable

This bit is set and cleared by software to enable/disable the master timer compare 1 interrupt

0: Compare 1 interrupt disabled

1: Compare 1 interrupt enabled

27.5.5 HRTIM master timer counter register (HRTIM_MCNTR)

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
MCNT[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 **MCNT[15:0]**: Counter value

Holds the master timer counter value. This register can only be written when the master timer is stopped (MCEN = 0 in HRTIM_MCR).

Note: For HR clock prescaling ratio below 32 (CKPSCCKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

Note: The timer behavior is not guaranteed if the counter value is set above the HRTIM_MPER register value.

27.5.6 HRTIM master timer period register (HRTIM_MPER)

Address offset: 0x014

Reset value: 0x0000 FFDF

31	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
MPER[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 **MPER[15:0]**: Master timer period value

This register defines the counter overflow value.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The maximum value is 0x0000 FFDF.

27.5.7 HRTIM master timer repetition register (HRTIM_MREP)

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.	MREP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **MREP[7:0]**: Master timer repetition period value

This register holds the repetition period value for the master counter. It is either the preload register or the active register if preload is disabled.

27.5.8 HRTIM master timer compare 1 register (HRTIM_MCMP1R)

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
MCMP1[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 **MCMP1[15:0]**: Master timer compare 1 value

This register holds the master timer compare 1 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

27.5.9 HRTIM master timer compare 2 register (HRTIM_MCMP2R)

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
MCMP2[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 **MCMP2[15:0]**: Master timer compare 2 value

This register holds the master timer compare 2 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

27.5.10 HRTIM master timer compare 3 register (HRTIM_MCMP3R)

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
MCMP3[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 **MCMP3[15:0]**: Master timer compare 3 value

This register holds the master timer compare 3 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

27.5.11 HRTIM master timer compare 4 register (HRTIM_MCMP4R)

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
MCMP4[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 **MCMP4[15:0]**: Master timer compare 4 value

This register holds the master timer compare 4 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

27.5.12 HRTIM timer x control register (HRTIM_TIMxCR) (x = A to F)

Address offset: Block A: 0x080

Address offset: Block B: 0x100

Address offset: Block C: 0x180

Address offset: Block D: 0x200

Address offset: Block E: 0x280

Address offset: Block F: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDGAT[3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	TAU	TxRSTU	TxREPU	TFU
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
DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	RSYNCU	INTLVD[1:0]		PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **UPDGAT[3:0]**: Update gating

These bits define how the update occurs relatively to the burst DMA transaction and the external update request on update enable inputs hrtim_upd_en[3:1] (see *Table 209*).

The update events, as mentioned below, can be: MSTU, TFU, TEU, TDU, TCU, TBU, TAU, TxRSTU, TxREPU.

0000: The update occurs independently from the DMA burst transfer
0001: The update occurs when the DMA burst transfer is completed
0010: The update occurs on the update event following the DMA burst transfer completion
0011: The update occurs on a rising edge on hrtim_upd_en1
0100: The update occurs on a rising edge on hrtim_upd_en2
0101: The update occurs on a rising edge on hrtim_upd_en3
0110: The update occurs on the update event following a rising edge on hrtim_upd_en1
0111: The update occurs on the update event following a rising edge on hrtim_upd_en2
1000: The update occurs on the update event following a rising edge on hrtim_upd_en3
Others: Reserved

Note: This bitfield must be reset before programming a new value.
For UPDGAT[3:0] values equal to 0001, 0011, 0100, 0101, it is possible to have multiple concurrent update source (for instance RSTU and DMA burst).

Bit 27 **PREEN**: Preload enable

This bit enables the registers preload mechanism and defines whether a write access into a preload-able register is done into the active or the preload register.

0: Preload disabled: the write access is directly done into the active register
1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]** DAC synchronization

A DAC synchronization event is generated when the timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 27.3.21: DAC triggers* for connections details).

00: No DAC trigger generated
01: Trigger generated on hrtim_dac_trg1
10: Trigger generated on hrtim_dac_trg2
11: Trigger generated on hrtim_dac_trg3

Bit 24 **MSTU**: Master timer update

Register update is triggered by the master timer update.

0: Update by master timer disabled
1: Update by master timer enabled

Bit 23 **TEU**: Timer E update

Register update is triggered by the timer E update

0: Update by timer E disabled
1: Update by timer E enabled

Note: This bit is reserved for HRTIM_TIMECR*. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMFCR.*

Bit 22 **TDU**: Timer D update

Register update is triggered by the timer D update

0: Update by timer D disabled
1: Update by timer D enabled

Note: This bit is reserved for HRTIM_TIMDCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 21 **TCU**: Timer C update

Register update is triggered by the timer C update
0: Update by timer C disabled
1: Update by timer C enabled

Note: This bit is reserved for HRTIM_TIMCCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 20 **TBU**: Timer B update

Register update is triggered by the timer B update
0: Update by timer B disabled
1: Update by timer B enabled

Note: This bit is reserved for HRTIM_TIMBCR. It is only available for HRTIM_TIMACR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 19 **TAU**: Timer A update

Register update is triggered by the timer A update
0: Update by timer A disabled
1: Update by timer A enabled

Note: This bit is reserved for HRTIM_TIMBCR. It is only available for HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 18 **TxRSTU**: Timer x reset update

Register update is triggered by timer x counter reset or roll-over to 0 after reaching the period value in continuous mode.
0: Update by timer x reset / roll-over disabled
1: Update by timer x reset / roll-over enabled

Bit 17 **TxREPU**: Timer x repetition update

Register update is triggered when the counter rolls over and HRTIM_REPx = 0
0: Update on repetition disabled
1: Update on repetition enabled

Bit 16 **TFU**: Timer F update

Register update is triggered by the timer F update
0: Update by timer F disabled
1: Update by timer F enabled

Note: This bit is reserved for HRTIM_TIMFCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR.

Bits 15:14 **DELCMP4[1:0]**: CMP4 auto-delayed mode

This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Section : Auto-delayed mode*).

00: CMP4 register is always active (standard compare mode)
01: CMP4 value is recomputed and is active following a capture 2 event
10: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 1 match (timeout function if capture 2 event is missing)
11: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 3 match (timeout function if capture event is missing)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bits 13:12 **DELCMP2[1:0]**: CMP2 auto-delayed mode

This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Section : Auto-delayed mode*).

00: CMP2 register is always active (standard compare mode)

01: CMP2 value is recomputed and is active following a capture 1 event

10: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after compare 1 match (timeout function if capture event is missing)

11: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after compare 3 match (timeout function if capture event is missing)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 11 **SYNCSTRTx**: Synchronization starts timer x

This bit defines the timer x behavior following the synchronization event:

0: No effect on timer x

1: A synchronization input event starts the timer x

Bit 10 **SYNCRSTx**: Synchronization resets timer x

This bit defines the timer x behavior following the synchronization event:

0: No effect on timer x

1: A synchronization input event resets the timer x

Bit 9 **RSYNCU**: Re-synchronized update

This bit specifies whether update source coming outside from the timing unit must be synchronized:

0: The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) is taken into account immediately

1: The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) is taken into account on the following reset/roll-over event.

Note: This bit is significant only when UPDGAT[3:0] = 0000, it is ignored otherwise.

Bits 8:7 **INTLVD[1:0]**: Interleaved mode

This bitfield is significant only when the HALF bit is reset. It enables the interleaved mode.

00: Interleaved mode disabled

01: Triple interleaved mode: when HRTIM_PERxR register is written, the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/3 value, and the HRTIM_CMP2xR active register is automatically updated with 2x (HRTIM_PERxR/3) value.

10: Quad interleaved mode: when HRTIM_PERxR register is written, the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/4 value, the HRTIM_CMP2xR active register is automatically updated with HRTIM_PERxR/2 value and the HRTIM_CMP3xR active register is automatically updated with 3x (HRTIM_PERxR/4) value.

11: Interleaved mode disabled

Bit 6 **PSHPLL**: Push-pull mode enable

This bit enables the push-pull mode.

0: Push-pull mode disabled

1: Push-pull mode enabled

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 5 **HALF**: Half mode enable

This bit enables the half duty-cycle mode: the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/2 value when HRTIM_PERxR register is written.

0: Half mode disabled

1: Half mode enabled

Bit 4 **RETRIG**: Re-triggerable mode

This bit defines the counter behavior in single shot mode.

0: The timer is not re-triggerable: a counter reset is done if the counter is stopped (period elapsed in single-shot mode or counter stopped in continuous mode)

1: The timer is re-triggerable: a counter reset is done whatever the counter state.

Bit 3 **CONT**: Continuous mode

This bit defines the timer operating mode.

0: The timer operates in single-shot mode and stops when it reaches TIMxPER value

1: The timer operates in continuous mode and rolls over to zero when it reaches TIMxPER value

Bits 2:0 **CKPSCx[2:0]**: HRTIM timer x clock prescaler

These bits define the master timer high-resolution clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to f_{HRCK} / $2^{CKPSC[2:0]}$.

The prescaling ratio cannot be modified once the timer is enabled.

27.5.13 HRTIM timer x interrupt status register (HRTIM_TIMxISR) (x = A to F)

Address offset: Block A: 0x084

Address offset: Block B: 0x104

Address offset: Block C: 0x184

Address offset: Block D: 0x204

Address offset: Block E: 0x284

Address offset: Block F: 0x304

Reset value: 0x000 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.	O2CPY	O1CPY	O2 STAT	O1 STAT	IPP STAT	CPP STAT
										r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	DLYPR T	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	Res.	REP	CMP4	CMP3	CMP2	CMP1
	r	r	r	r	r	r	r	r	r		r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **O2CPY**: Output 2 copy

This status bit is a raw copy of the output 2 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 2 is inactive

1: Output 2 is active

Bit 20 **O1CPY**: Output 1 copy

This status bit is a raw copy of the output 1 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 1 is inactive

1: Output 1 is active

Bit 19 **O2STAT**: Output 2 status

This status bit indicates the output 2 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 2 was inactive

1: Output 2 was active

Bit 18 **O1STAT**: Output 1 status

This status bit indicates the output 1 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 1 was inactive

1: Output 1 was active

Bit 17 **IPPSTAT**: Idle push-pull Status

This status bit indicates on which output the signal was applied, in push-pull mode balanced fault mode or delayed idle mode, when the protection was triggered (whatever the output state, active or inactive).

0: Protection occurred when the output 1 was active and output 2 forced inactive

1: Protection occurred when the output 2 was active and output 1 forced inactive

Bit 16 **CPPSTAT**: Current push-pull status

This status bit indicates on which output the signal is currently applied, in push-pull mode. It is only significant in this configuration.

0: Signal applied on output 1 and output 2 forced inactive

1: Signal applied on output 2 and output 1 forced inactive

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DLYPRT**: Delayed protection flag

0: No delayed protection interrupt occured

1: Delayed Idle or balanced Idle mode entry occured

Bit 13 **RST**: Reset and/or roll-over interrupt flag

This bit is set by hardware when the timer x counter is reset or rolls over in continuous mode.

0: No TIMx counter reset/roll-over interrupt occurred

1: TIMX counter reset/roll-over interrupt occurred

Bit 12 **RSTx2**: Output 2 reset interrupt flag

Refer to RSTx1 description

Bit 11 **SETx2**: Output 2 set interrupt flag

Refer to SETx1 description

Bit 10 **RSTx1**: Output 1 reset interrupt flag

This bit is set by hardware when the Tx1 output is reset (goes from active to inactive mode).

0: No Tx1 output reset interrupt occurred

1: Tx1 output reset interrupt occurred

Bit 9 **SETx1**: Output 1 set interrupt flag

This bit is set by hardware when the Tx1 output is set (goes from inactive to active mode).

0: No Tx1 output set interrupt occurred

1: Tx1 output set interrupt occurred

Bit 8 **CPT2**: Capture 2 interrupt flag

Refer to CPT1 description

Bit 7 **CPT1**: Capture 1 interrupt flag

This bit is set by hardware when the timer x capture 1 event occurs.

0: No timer x capture 1 interrupt occurred

1: Timer x capture 1 interrupt occurred

Bit 6 **UPD**: Update interrupt flag

This bit is set by hardware when the timer x update event occurs.

0: No timer x update interrupt occurred

1: Timer x update interrupt occurred

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REP**: Repetition interrupt flag

This bit is set by hardware when the timer x repetition period has elapsed.

0: No timer x repetition interrupt occurred

1: Timer x repetition interrupt occurred

Bit 3 **CMP4**: Compare 4 interrupt flag

Refer to CMP1 description

Bit 2 **CMP3**: Compare 3 interrupt flag

Refer to CMP1 description

Bit 1 **CMP2**: Compare 2 interrupt flag

Refer to CMP1 description

Bit 0 **CMP1**: Compare 1 interrupt flag

This bit is set by hardware when the timer x counter matches the value programmed in the compare 1 register.

0: No compare 1 interrupt occurred

1: Compare 1 interrupt occurred

27.5.14 HRTIM timer x interrupt clear register (HRTIM_TIMxICR) (x = A to F)

Address offset: Block A: 0x088

Address offset: Block B: 0x108

Address offset: Block C: 0x188

Address offset: Block D: 0x208

Address offset: Block E: 0x288

Address offset: Block F: 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.	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.	DLYPRTC	RSTC	RSTx2C	SET2xC	RSTx1C	SET1xC	CPT2C	CPT1C	UPDC	Res.	REPC	CMP4C	CMP3C	CMP2C	CMP1C
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **DLYPRTC**: Delayed protection flag clear

Writing 1 to this bit clears the DLYPRT flag in HRTIM_TIMxISR register

Bit 13 **RSTC**: Reset interrupt flag clear

Writing 1 to this bit clears the RST flag in HRTIM_TIMxISR register

Bit 12 **RSTx2C**: Output 2 reset flag clear

Writing 1 to this bit clears the RSTx2 flag in HRTIM_TIMxISR register

Bit 11 **SETx2C**: Output 2 set flag clear

Writing 1 to this bit clears the SETx2 flag in HRTIM_TIMxISR register

Bit 10 **RSTx1C**: Output 1 reset flag clear

Writing 1 to this bit clears the RSTx1 flag in HRTIM_TIMxISR register

Bit 9 **SETx1C**: Output 1 set flag clear

Writing 1 to this bit clears the SETx1 flag in HRTIM_TIMxISR register

Bit 8 **CPT2C**: Capture 2 interrupt flag clear

Writing 1 to this bit clears the CPT2 flag in HRTIM_TIMxISR register

Bit 7 **CPT1C**: Capture 1 interrupt flag clear

Writing 1 to this bit clears the CPT1 flag in HRTIM_TIMxISR register

Bit 6 **UPDC**: Update interrupt flag clear

Writing 1 to this bit clears the UPD flag in HRTIM_TIMxISR register

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REPC**: Repetition interrupt flag clear

Writing 1 to this bit clears the REP flag in HRTIM_TIMxISR register

Bit 3 **CMP4C**: Compare 4 interrupt flag clear

Writing 1 to this bit clears the CMP4 flag in HRTIM_TIMxISR register

Bit 2 **CMP3C**: Compare 3 interrupt flag clear

Writing 1 to this bit clears the CMP3 flag in HRTIM_TIMxISR register

Bit 1 **CMP2C**: Compare 2 interrupt flag clear

Writing 1 to this bit clears the CMP2 flag in HRTIM_TIMxISR register

Bit 0 **CMP1C**: Compare 1 interrupt flag clear

Writing 1 to this bit clears the CMP1 flag in HRTIM_TIMxISR register

27.5.15 HRTIM timer x DMA interrupt enable register (HRTIM_TIMxDIER) (x = A to F)

Address offset: Block A: 0x08C

Address offset: Block B: 0x10C

Address offset: Block C: 0x18C

Address offset: Block D: 0x20C

Address offset: Block E: 0x28C

Address offset: Block F: 0x30C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SETx1DE	CPT2DE	CPT1DE	UPDDE	Res.	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE
	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.	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SET1xIE	CPT2IE	CPT1IE	UPDIE	Res.	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
	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 **DLYPRTDE**: Delayed protection DMA request enable

This bit is set and cleared by software to enable/disable DMA requests on delayed protection.

0: Delayed protection DMA request disabled

1: Delayed protection DMA request enabled

Bit 29 **RSTDE**: Reset/roll-over DMA request enable

This bit is set and cleared by software to enable/disable DMA requests on timer x counter reset or roll-over in continuous mode.

0: Timer x counter reset/roll-over DMA request disabled

1: Timer x counter reset/roll-over DMA request enabled

Bit 28 **RSTx2DE**: Output 2 reset DMA request enable

Refer to RSTx1DE description

Bit 27 **SETx2DE**: Output 2 set DMA request enable

Refer to SETx1DE description

Bit 26 **RSTx1DE**: Output 1 reset DMA request enable

This bit is set and cleared by software to enable/disable Tx1 output reset DMA requests.

0: Tx1 output reset DMA request disabled

1: Tx1 output reset DMA request enabled

Bit 25 **SETx1DE**: Output 1 set DMA request enable

This bit is set and cleared by software to enable/disable Tx1 output set DMA requests.

0: Tx1 output set DMA request disabled

1: Tx1 output set DMA request enabled

Bit 24 **CPT2DE**: Capture 2 DMA request enable

Refer to CPT1DE description

Bit 23 **CPT1DE**: Capture 1 DMA request enable

This bit is set and cleared by software to enable/disable capture 1 DMA requests.

0: Capture 1 DMA request disabled

1: Capture 1 DMA request enabled

Bit 22 **UPDDE**: Update DMA request enable

This bit is set and cleared by software to enable/disable DMA requests on update event.

0: Update DMA request disabled

1: Update DMA request enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **REPDE**: Repetition DMA request enable

This bit is set and cleared by software to enable/disable DMA requests on repetition event.

0: Repetition DMA request disabled

1: Repetition DMA request enabled

Bit 19 **CMP4DE**: Compare 4 DMA request enable

Refer to CMP1DE description

Bit 18 **CMP3DE**: Compare 3 DMA request enable

Refer to CMP1DE description

Bit 17 **CMP2DE**: Compare 2 DMA request enable

Refer to CMP1DE description

Bit 16 **CMP1DE**: Compare 1 DMA request enable

This bit is set and cleared by software to enable/disable the compare 1 DMA requests.

0: Compare 1 DMA request disabled

1: Compare 1 DMA request enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DLYPRTIE**: Delayed protection interrupt enable

This bit is set and cleared by software to enable/disable interrupts on delayed protection.

0: Delayed protection interrupts disabled

1: Delayed protection interrupts enabled

Bit 13 **RSTIE**: Reset/roll-over interrupt enable

This bit is set and cleared by software to enable/disable interrupts on timer x counter reset or roll-over in continuous mode.

0: Timer x counter reset/roll-over interrupt disabled

1: Timer x counter reset/roll-over interrupt enabled

Bit 12 **RSTx2IE**: Output 2 reset interrupt enable

Refer to RSTx1IE description

Bit 11 **SETx2IE**: Output 2 set interrupt enable

Refer to SETx1IE description

Bit 10 **RSTx1IE**: Output 1 reset interrupt enable

This bit is set and cleared by software to enable/disable Tx1 output reset interrupts.

0: Tx1 output reset interrupts disabled

1: Tx1 output reset interrupts enabled

Bit 9 **SETx1IE**: Output 1 set interrupt enable

This bit is set and cleared by software to enable/disable Tx1 output set interrupts.

0: Tx1 output set interrupts disabled

1: Tx1 output set interrupts enabled

Bit 8 **CPT2IE**: Capture interrupt enable

Refer to CPT1IE description

Bit 7 **CPT1IE**: Capture interrupt enable

This bit is set and cleared by software to enable/disable capture 1 interrupts.

0: Capture 1 interrupts disabled

1: Capture 1 interrupts enabled

Bit 6 **UPDIE**: Update interrupt enable

This bit is set and cleared by software to enable/disable update event interrupts.

0: Update interrupts disabled

1: Update interrupts enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REPIE**: Repetition interrupt enable

This bit is set and cleared by software to enable/disable repetition event interrupts.

0: Repetition interrupts disabled

1: Repetition interrupts enabled

Bit 3 **CMP4IE**: Compare 4 interrupt enable

Refer to CMP1IE description

Bit 2 **CMP3IE**: Compare 3 interrupt enable
Refer to CMP1IE description

Bit 1 **CMP2IE**: Compare 2 interrupt enable
Refer to CMP1IE description

Bit 0 **CMP1IE**: Compare 1 interrupt enable
This bit is set and cleared by software to enable/disable the compare 1 interrupts.
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

27.5.16 HRTIM timer x counter register (HRTIM_CNTxR) (x = A to F)

Address offset: Block A: 0x090

Address offset: Block B: 0x110

Address offset: Block C: 0x190

Address offset: Block D: 0x210

Address offset: Block E: 0x290

Address offset: Block F: 0x310

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
CNTx[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 **CNTx[15:0]**: Timer x counter value

This register holds the timer x counter value. It can only be written when the timer is stopped (TxCEN = 0 in HRTIM_TIMxCR).

Note: For HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

Note: The timer behavior is not guaranteed if the counter value is above the HRTIM_PERxR register value.

27.5.17 HRTIM timer x period register (HRTIM_PERxR) (x = A to F)

Address offset: Block A: 0x094

Address offset: Block B: 0x114

Address offset: Block C: 0x194

Address offset: Block D: 0x214

Address offset: Block E: 0x294

Address offset: Block F: 0x314

Reset value: 0x0000 FFDF

31	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
PERx[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 **PERx[15:0]**: Timer x period value

This register holds timer x period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The maximum value is 0x0000 FFDF.

27.5.18 HRTIM timer x repetition register (HRTIM_REPxR) (x = A to F)

Address offset: Block A: 0x098

Address offset: Block B: 0x118

Address offset: Block C: 0x198

Address offset: Block D: 0x218

Address offset: Block E: 0x298

Address offset: Block F: 0x318

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.	REPx[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits31:8 Reserved, must be kept at reset value.

Bits 7:0 **REPx[7:0]**: Timer x repetition period value

This register holds the repetition period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

27.5.19 HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F)

Address offset: Block A: 0x09C

Address offset: Block B: 0x11C

Address offset: Block C: 0x19C

Address offset: Block D: 0x21C

Address offset: Block E: 0x29C

Address offset: Block F: 0x31C

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
CMP1x[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 **CMP1x[15:0]**: Timer x compare 1 value

This register holds the compare 1 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

27.5.20 HRTIM timer x compare 1 compound register (HRTIM_CMP1CxR) (x = A to F)

Address offset: Block A: 0x0A0

Address offset: Block B: 0x120

Address offset: Block C: 0x1A0

Address offset: Block D: 0x220

Address offset: Block E: 0x2A0

Address offset: Block F: 0x320

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.	REPx[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[15:0]															
rw	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 23:16 **REPx[7:0]**: Timer x repetition value (aliased from HRTIM_REPx register)
This bitfield is an alias from the REPx[7:0] bitfield in the HRTIMx_REPxR register.

Bits 15:0 **CMP1x[15:0]**: Timer x compare 1 value
This bitfield is an alias from the CMP1x[15:0] bitfield in the HRTIMx_CMP1xR register.

27.5.21 HRTIM timer x compare 2 register (HRTIM_CMP2xR) (x = A to F)

Address offset: Block A: 0x0A4

Address offset: Block B: 0x124

Address offset: Block C: 0x1A4

Address offset: Block D: 0x224

Address offset: Block E: 0x2A4

Address offset: Block F: 0x324

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
CMP2x[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 **CMP2x[15:0]**: Timer x compare 2 value

This register holds the compare 2 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

This register behaves as an auto-delayed compare register, if enabled with DELCMP2[1:0] bits in HRTIM_TIMxCR.

27.5.22 HRTIM timer x compare 3 register (HRTIM_CMP3xR) (x = A to F)

Address offset: Block A: 0x0A8

Address offset: Block B: 0x128

Address offset: Block C: 0x1A8

Address offset: Block D: 0x228

Address offset: Block E: 0x2A8

Address offset: Block F: 0x328

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
CMP3x[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 **CMP3x[15:0]**: Timer x compare 3 value

This register holds the compare 3 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the fHRTIM clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

27.5.23 HRTIM timer x compare 4 register (HRTIM_CMP4xR) (x = A to F)

Address offset: Block A: 0x0AC

Address offset: Block B: 0x12C

Address offset: Block C: 0x1AC

Address offset: Block D: 0x22C

Address offset: Block E: 0x2AC

Address offset: Block F: 0x32C

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
CMP4x[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 **CMP4x[15:0]**: Timer x compare 4 value

This register holds the compare 4 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

This register can behave as an auto-delayed compare register, if enabled with DELCMP4[1:0] bits in HRTIM_TIMxCR.

27.5.24 HRTIM timer x capture 1 register (HRTIM_CPT1xR) (x = A to F)

Address offset: Block A: 0x0B0

Address offset: Block B: 0x130

Address offset: Block C: 0x1B0

Address offset: Block D: 0x230

Address offset: Block E: 0x2B0

Address offset: Block F: 0x330

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.	DIR
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT1x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 16 **DIR**: Timer x capture 1 direction status

This register holds the counting direction value when the capture 1 event occurred:
0: timer is up-counting
1: timer is down-counting
In up-counting mode (UDM bit reset), the DIR bit is always read as 0.

Bits 15:0 **CPT1x[15:0]**: Timer x capture 1 value

This register holds the counter value when the capture 1 event occurred.

Note: In up/down mode (UDM bit set to 1), the capture value is referred to:
- counting reset, when up-counting
- the PER event when down counting

The DIR bit allows to discriminate the up-down phases when reading the captured value.

Note: This is a regular resolution register: for HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

27.5.25 HRTIM timer x capture 2 register (HRTIM_CPT2xR) (x = A to F)

Address offset: Block A: 0x0B4

Address offset: Block B: 0x134

Address offset: Block C: 0x1B4

Address offset: Block D: 0x234

Address offset: Block E: 0x2B4

Address offset: Block F: 0x334

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.	DIR
															r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CPT2x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 16 **DIR**: Timer x capture 1 direction status

This register holds the counting direction value when the capture 1 event occurred:
0: timer is up-counting
1: timer is down-counting
In up-counting mode (UDM bit reset), the DIR bit is always read as 0.

Bits 15:0 **CPT2x[15:0]**: Timer x capture 2 value

This register holds the counter value when the capture 2 event occurred.

Note: In up/down mode (UDM bit set to 1), the capture value is referred to:
- counting reset, when up-counting
- the PER event when down counting

The DIR bit allows to discriminate the up-down phases when reading the captured value.

Note: This is a regular resolution register: for HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

27.5.26 HRTIM timer x deadtime register (HRTIM_DTxR) (x = A to F)

Address offset: Block A: 0x0B8

Address offset: Block B: 0x138

Address offset: Block C: 0x1B8

Address offset: Block D: 0x238

Address offset: Block E: 0x2B8

Address offset: Block F: 0x338

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTFLKx	DTFSLKx	Res.	Res.	Res.	Res.	SDTFx	DTFx[8:0]								
rwo	rwo					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
DTRLKx	DTRSLKx	Res.	DTPRSC[1:0]			SDTRx	DTRx[8:0]								
rwo	rwo		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DTFLKx**: Deadtime falling lock

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.

0: Deadtime falling value and sign is writable

1: Deadtime falling value and sign is read-only

Note: This bit is not preloaded

Bit 30 **DTFSLKx**: Deadtime falling sign lock

This write-once bit prevents the sign of falling deadtime to be modified, if enabled.

0: Deadtime falling sign is writable

1: Deadtime falling sign is read-only

Note: This bit is not preloaded.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **SDTFx**: Sign deadtime falling value

This register determines whether the deadtime is positive (signals not overlapping) or negative (signals overlapping).

0: Positive deadtime on falling edge

1: Negative deadtime on falling edge

Bits 24:16 **DTFx[8:0]**: Deadtime falling value

This register holds the value of the deadtime following a falling edge of reference PWM signal.

t_{DTF} = DTFx[8:0] x t_{DTG}

Bit 15 **DTRLKx**: Deadtime rising lock

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.

0: Deadtime rising value and sign is writable

1: Deadtime rising value and sign is read-only

Note: This bit is not preloaded.

Bit 14 **DTRSLKx**: Deadtime rising sign lock

This write-once bit prevents the sign of deadtime to be modified, if enabled.

0: Deadtime rising sign is writable

1: Deadtime rising sign is read-only

Note: This bit is not preloaded.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DTPRSC[2:0]**: Deadtime prescaler

This register holds the value of the deadtime clock prescaler.

$t_{DTG} = (2^{(DTPRSC[2:0])}) \times (t_{HRTIM} / 8)$

(i.e. 000: 735 ps, 001= 1.471 ns,...)

This bitfield is read-only as soon as any of the lock bit is enabled (DTFLKs, DTFSLKx, DTRLKx, DTRSLKx).

Bit 9 **SDTRx**: Sign deadtime rising value

This register determines whether the deadtime is positive or negative (overlapping signals).

0: Positive deadtime on rising edge

1: Negative deadtime on rising edge

Bits 8:0 **DTRx[8:0]**: Deadtime rising value

This register holds the value of the deadtime following a rising edge of reference PWM signal.

$t_{DTR} = DTRx[8:0] \times t_{DTG}$

27.5.27 HRTIM timer x output 1 set register (HRTIM_SETx1R) (x = A to F)

Address offset: Block A: 0x0BC

Address offset: Block B: 0x13C

Address offset: Block C: 0x1BC

Address offset: Block D: 0x23C

Address offset: Block E: 0x2BC

Address offset: Block F: 0x33C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXT EVNT10	EXT EVNT9	EXT EVNT8	EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	TIM EVNT9	TIM EVNT8	TIM EVNT7	TIM EVNT6	TIM EVNT5
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**
TIM EVNT4	TIM EVNT3	TIM EVNT2	TIM EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer preload to active)
Register update event forces the output to its active state.

Bit 30 **EXTEVNT10**: External event 10
Refer to EXTEVNT1 description.

Bit 29 **EXTEVNT9**: External event 9
Refer to EXTEVNT1 description.

Bit 28 **EXTEVNT8**: External event 8
Refer to EXTEVNT1 description.

Bit 27 **EXTEVNT7**: External event 7
Refer to EXTEVNT1 description.

Bit 26 **EXTEVNT6**: External event 6
Refer to EXTEVNT1 description.

Bit 25 **EXTEVNT5**: External event 5
Refer to EXTEVNT1 description.

Bit 24 **EXTEVNT4**: External event 4
Refer to EXTEVNT1 description.

Bit 23 **EXTEVNT3**: External event 3
Refer to EXTEVNT1 description.

Bit 22 **EXTEVNT2**: External event 2
Refer to EXTEVNT1 description.

Bit 21 **EXTEVNT1**: External event 1
External event 1 forces the output to its active state.

Bit 20 **TIMEVNT9**: Timer event 9
Refer to TIMEVNT1 description.

Bit 19 **TIMEVNT8**: Timer event 8
Refer to TIMEVNT1 description.

Bit 18 **TIMEVNT7**: Timer event 7
Refer to TIMEVNT1 description.

Bit 17 **TIMEVNT6**: Timer event 6
Refer to TIMEVNT1 description.

Bit 16 **TIMEVNT5**: Timer event 5
Refer to TIMEVNT1 description.

Bit 15 **TIMEVNT4**: Timer event 4
Refer to TIMEVNT1 description.

Bit 14 **TIMEVNT3**: Timer event 3
Refer to TIMEVNT1 description.

Bit 13 **TIMEVNT2**: Timer event 2
Refer to TIMEVNT1 description.

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its active state (refer to *Table 216* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its active state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its active state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its active state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its active state.

Bit 7 **MSTPER**: Master period
The master timer counter roll-over in continuous mode, or to the master timer reset in single-shot mode forces the output to its active state.

Bit 6 **CMP4**: Timer x compare 4
Timer A compare 4 event forces the output to its active state.

Bit 5 **CMP3**: Timer x compare 3
Timer A compare 3 event forces the output to its active state.

Bit 4 **CMP2**: Timer x compare 2
Timer A compare 2 event forces the output to its active state.

Bit 3 **CMP1**: Timer x compare 1
Timer A compare 1 event forces the output to its active state.

Bit 2 **PER**: Timer x period

Timer A period event forces the output to its active state.

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer A resynchronization

Timer A reset event coming solely from software or SYNC input forces the output to its active state.

Note: Other timer reset are not affecting the output when RESYNC=1.

Bit 0 **SST**: Software set trigger

This bit forces the output to its active state. This bit can only be set by software and is reset by hardware.

Note: This bit is not preloaded.

27.5.28 HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F)

Address offset: Block A: 0x0C0

Address offset: Block B: 0x140

Address offset: Block C: 0x1C0

Address offset: Block D: 0x240

Address offset: Block E: 0x2C0

Address offset: Block F: 0x340

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXT EVNT10	EXT EVNT9	EXT EVNT8	EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	TIM EVNT9	TIM EVNT8	TIM EVNT7	TIM EVNT6	TIM EVNT5
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**
TIM EVNT4	TIM EVNT3	TIM EVNT2	TIM EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.
These bits are defining the source which can force the Tx1 output to its inactive state.

27.5.29 HRTIM timer x output 2 set register (HRTIM_SETx2R) (x = A to F)

Address offset: Block A: 0x0C4

Address offset: Block B: 0x144

Address offset: Block C: 0x1C4

Address offset: Block D: 0x244

Address offset: Block E: 0x2C4

Address offset: Block F: 0x344

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXT EVNT10	EXT EVNT9	EXT EVNT8	EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	TIM EVNT9	TIM EVNT8	TIM EVNT7	TIM EVNT6	TIM EVNT5
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**
TIM EVNT4	TIM EVNT3	TIM EVNT2	TIM EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.
These bits are defining the source which forces the Tx2 output to its active state.

27.5.30 HRTIM timer x output 2 reset register (HRTIM_RSTx2R) (x = A to F)

Address offset: Block A: 0x0C8

Address offset: Block B: 0x148

Address offset: Block C: 0x1C8

Address offset: Block D: 0x248

Address offset: Block E: 0x2C8

Address offset: Block F: 0x348

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXT EVNT10	EXT EVNT9	EXT EVNT8	EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	TIM EVNT9	TIM EVNT8	TIM EVNT7	TIM EVNT6	TIM EVNT5
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
TIM EVNT4	TIM EVNT3	TIM EVNT2	TIM EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.
These bits are defining the source which forces the Tx2 output to its inactive state.

27.5.31 HRTIM timer x external event filtering register 1 (HRTIM_EEFxR1) (x = A to F)

Address offset: Block A: 0x0CC

Address offset: Block B: 0x14C

Address offset: Block C: 0x1CC

Address offset: Block D: 0x24C

Address offset: Block E: 0x2CC

Address offset: Block F: 0x34C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE5FLTR[3:0]				EE5LTCH	Res.	EE4FLTR[3:0]				EE4LTCH	Res.	EE3FLTR[3]
			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
EE3FLTR[2:0]			EE3LTCH	Res.	EE2FLTR[3:0]				EE2LTCH	Res.	EE1FLTR[3:0]				EE1LTCH
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **EE5FLTR[3:0]**: External event 5 filter
Refer to EE1FLTR[3:0] description.

Bit 24 **EE5LTCH**: External event 5 latch
Refer to EE1LTCH description

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 **EE4FLTR[3:0]**: External event 4 filter
Refer to EE1FLTR[3:0] description.

Bit 18 **EE4LTCH**: External event 4 latch
Refer to EE1LTCH description

Bit 17 Reserved, must be kept at reset value.

Bits 16:13 **EE3FLTR[3:0]**: External event 3 filter
Refer to EE1FLTR[3:0] description.

Bit 12 **EE3LTCH**: External event 3 latch
Refer to EE1LTCH description

Bit 11 Reserved, must be kept at reset value.

Bits 10:7 **EE2FLTR[3:0]**: External event 2 filter
Refer to EE1FLTR[3:0] description.

Bit 6 **EE2LTCH**: External event 2 latch
Refer to EE1LTCH description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:1 **EE1FLTR[3:0]**: External event 1 filter

0000: No filtering
0001: Blanking from counter reset/roll-over to compare 1
0010: Blanking from counter reset/roll-over to compare 2 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): blanking from compare 1 to compare 2, only during the up-counting phase.
0011: Blanking from counter reset/roll-over to compare 3
0100: Blanking from counter reset/roll-over to compare 4
0100: Blanking from counter reset/roll-over to compare 4 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): blanking from compare 3 to compare 4, only during the up-counting phase.
0101: Blanking from another timing unit: TIMFLTR1 source (see *Table 224* for details)
0110: Blanking from another timing unit: TIMFLTR2 source (see *Table 224* for details)
0111: Blanking from another timing unit: TIMFLTR3 source (see *Table 224* for details)
1000: Blanking from another timing unit: TIMFLTR4 source (see *Table 224* for details)
1001: Blanking from another timing unit: TIMFLTR5 source (see *Table 224* for details)
1010: Blanking from another timing unit: TIMFLTR6 source (see *Table 224* for details)
1011: Blanking from another timing unit: TIMFLTR7 source (see *Table 224* for details)
1100: Blanking from another timing unit: TIMFLTR8 source (see *Table 224* for details)
1101: Windowing from counter reset/roll-over to compare 2 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 to compare 3, only during the up-counting phase.
1110: Windowing from counter reset/roll-over to compare 3 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 to compare 3, only during the down-counting phase.
1111: Windowing from another timing unit: TIMWIN source (see *Table 225* for details) in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 during the up-counting phase to compare 3 during the down-counting phase.

Note: Whenever a compare register is used for filtering, the value must be strictly above 0.

This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 0 **EE1LTCH**: External event 1 latch

0: Event 1 is ignored if it happens during a blank, or passed through during a window.
1: Event 1 is latched and delayed till the end of the blanking or windowing period.

Note: A timeout event is generated in window mode (EE1FLTR[3:0]=1101, 1110, 1111) if EE1LTCH = 0, except if the external event is programmed in fast mode (EExFAST = 1).

This bitfield must not be modified once the counter is enabled (TxCEN bit set).

27.5.32 HRTIM timer x external event filtering register 2 (HRTIM_EEFxR2) (x = A to F)

Address offset: Block A: 0x0D0

Address offset: Block B: 0x150

Address offset: Block C: 0x1D0

Address offset: Block D: 0x250

Address offset: Block E: 0x2D0

Address offset: Block F: 0x350

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10FLTR[3:0]				EE10LTCH	Res.	EE9FLTR[3:0]				EE9LTCH	Res.	EE8FLTR[3]
			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**
EE8FLTR[2:0]			EE8LTCH	Res.	EE7FLTR[3:0]				EE7LTCH	Res.	EE6FLTR[3:0]				EE6LTCH
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **EE10FLTR[3:0]**: External event 10 filter
Refer to EE1FLTR[3:0] description.

Bit 24 **EE10LTCH**: External event 10 latch
Refer to EE1LTCH description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 **EE9FLTR[3:0]**: External event 9 filter
Refer to EE1FLTR[3:0] description.

Bit 18 **EE9LTCH**: External event 9 latch
Refer to EE1LTCH description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:13 **EE8FLTR[3:0]**: External event 8 filter
Refer to EE1FLTR[3:0] description.

Bit 12 **EE8LTCH**: External event 8 latch
Refer to EE1LTCH description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:7 **EE7FLTR[3:0]**: External event 7 filter
Refer to EE1FLTR[3:0] description.

Bit 6 **EE7LTCH**: External event 7 latch
Refer to EE1LTCH description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:1 **EE6FLTR[3:0]**: External event 6 filter
Refer to EE1FLTR[3:0] description.

Bit 0 **EE6LTCH**: External event 6 latch
Refer to EE1LTCH description.

27.5.33 HRTIM timer A reset register (HRTIM_RSTAR)

Address offset: 0x0D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIME CMP4	TIME CMP2	TIME CMP1	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
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
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2
The timer A counter is reset upon timer F Compare 2 event.

Bit 30 TIME**CPM4**: Timer E compare 4
The timer A counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2
The timer A counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1
The timer A counter is reset upon timer E compare 1 event.

Bit 27 **TIMDCMP4**: Timer D compare 4
The timer A counter is reset upon timer D compare 4 event.

Bit 26 **TIMDCMP2**: Timer D compare 2
The timer A counter is reset upon timer D compare 2 event.

Bit 25 **TIMDCMP1**: Timer D compare 1
The timer A counter is reset upon timer D compare 1 event.

Bit 24 **TIMCCMP4**: Timer C compare 4
The timer A counter is reset upon timer C compare 4 event.

Bit 23 **TIMCCMP2**: Timer C compare 2
The timer A counter is reset upon timer C compare 2 event.

Bit 22 **TIMCCMP1**: Timer C compare 1
The timer A counter is reset upon timer C compare 1 event.

Bit 21 **TIMBCMP4**: Timer B compare 4
The timer A counter is reset upon timer B compare 4 event.

Bit 20 **TIMBCMP2**: Timer B compare 2
The timer A counter is reset upon timer B compare 2 event.

Bit 19 **TIMBCMP1**: Timer B compare 1
The timer A counter is reset upon timer B compare 1 event.

Bit 18 **EXTEVNT10**: External event
The timer A counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9
The timer A counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer A counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer A counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer A counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer A counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer A counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer A counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer A counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer A counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer A counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer A counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer A counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer A counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer A counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer A compare 4 reset
The timer A counter is reset upon timer A compare 4 event.

Bit 2 **CMP2**: Timer A compare 2 reset
The timer A counter is reset upon timer A compare 2 event.

Bit 1 **UPDT**: Timer A update reset
The timer A counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer A counter is reset upon timer F compare 1 event.

27.5.34 HRTIM timer B reset register (HRTIM_RSTBR)

Address offset: 0x154

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIME CMP4	TIME CMP2	TIME CMP1	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
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**
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2

The timer B counter is reset upon timer F Compare 2 event.

Bit 30 TIME**CPM4**: Timer E compare 4

The timer B counter is reset upon timer E compare 4 event.

Bit 29 **T**IM**ECMP2**: Timer E compare 2

The timer B counter is reset upon timer E compare 2 event.

Bit 28 **T**IM**ECMP1**: Timer E compare 1

The timer B counter is reset upon timer E compare 1 event.

Bit 27 **T**IM**DCMP4**: Timer D compare 4

The timer B counter is reset upon timer D compare 4 event.

Bit 26 **T**IM**DCMP2**: Timer D compare 2

The timer B counter is reset upon timer D compare 2 event.

Bit 25 **T**IM**DCMP1**: Timer D compare 1

The timer B counter is reset upon timer D compare 1 event.

Bit 24 **T**IM**CCMP4**: Timer C compare 4

The timer B counter is reset upon timer C compare 4 event.

Bit 23 **T**IM**CCMP2**: Timer C compare 2

The timer B counter is reset upon timer C compare 2 event.

Bit 22 **T**IM**CCMP1**: Timer C compare 1

The timer B counter is reset upon timer C compare 1 event.

Bit 21 **T**IMA**CMP4**: Timer A compare 4

The timer B counter is reset upon timer A compare 4 event.

Bit 20 **T**IMA**CMP2**: Timer A compare 2

The timer B counter is reset upon timer A compare 2 event.

Bit 19 **T**IMA**CMP1**: Timer A compare 1

The timer B counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event

The timer B counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9

The timer B counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer B counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer B counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer B counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer B counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer B counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer B counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer B counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer B counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer B counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer B counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer B counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer B counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer B counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer B compare 4 reset
The timer B counter is reset upon timer B compare 4 event.

Bit 2 **CMP2**: Timer B compare 2 reset
The timer B counter is reset upon timer B compare 2 event.

Bit 1 **UPDT**: Timer B update reset
The timer B counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer B counter is reset upon timer F compare 1 event.

27.5.35 HRTIM timer C reset register (HRTIM_RSTCR)

Address offset: 0x2D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIME CMP4	TIME CMP2	TIME CMP1	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
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
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2

The timer C counter is reset upon timer F Compare 2 event.

Bit 30 TIME**CPM4**: Timer E compare 4

The timer C counter is reset upon timer E compare 4 event.

Bit 29 **T**IM**ECMP2**: Timer E compare 2

The timer C counter is reset upon timer E compare 2 event.

Bit 28 **T**IM**ECMP1**: Timer E compare 1

The timer C counter is reset upon timer E compare 1 event.

Bit 27 **T**IM**DCMP4**: Timer D compare 4

The timer C counter is reset upon timer D compare 4 event.

Bit 26 **T**IM**DCMP2**: Timer D compare 2

The timer C counter is reset upon timer D compare 2 event.

Bit 25 **T**IM**DCMP1**: Timer D compare 1

The timer C counter is reset upon timer D compare 1 event.

Bit 24 **T**IM**BCMP4**: Timer B compare 4

The timer C counter is reset upon timer B compare 4 event.

Bit 23 **T**IM**BCMP2**: Timer B compare 2

The timer C counter is reset upon timer B compare 2 event.

Bit 22 **T**IM**BCMP1**: Timer B compare 1

The timer C counter is reset upon timer B compare 1 event.

Bit 21 **T**IMA**CMP4**: Timer A compare 4

The timer C counter is reset upon timer A compare 4 event.

Bit 20 **T**IMA**CMP2**: Timer A compare 2

The timer C counter is reset upon timer A compare 2 event.

Bit 19 **T**IMA**CMP1**: Timer A compare 1

The timer C counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event

The timer C counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9

The timer C counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8

The timer C counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7

The timer C counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6

The timer C counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5

The timer C counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4

The timer C counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3

The timer C counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2

The timer C counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1

The timer C counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4

The timer C counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3

The timer C counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2

The timer C counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1

The timer C counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period

The timer C counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer C compare 4 reset

The timer C counter is reset upon timer C compare 4 event.

Bit 2 **CMP2**: Timer C compare 2 reset

The timer C counter is reset upon timer C compare 2 event.

Bit 1 **UPDT**: Timer C update reset

The timer C counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1

The timer C counter is reset upon timer F compare 1 event.

27.5.36 HRTIM timer D reset register (HRTIM_RSTDR)

Address offset: 0x254

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIME CMP4	TIME CMP2	TIME CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
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
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2

The timer D counter is reset upon timer F Compare 2 event.

Bit 30 TIME**CPM4**: Timer E compare 4

The timer D counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2

The timer D counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1

The timer D counter is reset upon timer E compare 1 event.

Bit 27 **TIMCCMP4**: Timer C compare 4

The timer D counter is reset upon timer C compare 4 event.

Bit 26 **TIMCCMP2**: Timer C compare 2

The timer D counter is reset upon timer C compare 2 event.

Bit 25 **TIMCCMP1**: Timer C compare 1

The timer D counter is reset upon timer C compare 1 event.

Bit 24 **TIMBCMP4**: Timer B compare 4

The timer D counter is reset upon timer B compare 4 event.

Bit 23 **TIMBCMP2**: Timer B compare 2

The timer D counter is reset upon timer B compare 2 event.

Bit 22 **TIMBCMP1**: Timer B compare 1

The timer D counter is reset upon timer B compare 1 event.

Bit 21 TIMA**CMP4**: Timer A compare 4

The timer D counter is reset upon timer A compare 4 event.

Bit 20 TIMA**CMP2**: Timer A compare 2

The timer D counter is reset upon timer A compare 2 event.

Bit 19 TIMA**CMP1**: Timer A compare 1

The timer D counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event

The timer D counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9

The timer D counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8

The timer D counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7

The timer D counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6

The timer D counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5

The timer D counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4

The timer D counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3

The timer D counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2

The timer D counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1

The timer D counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4

The timer D counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3

The timer D counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2

The timer D counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1

The timer D counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period

The timer D counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer D compare 4 reset

The timer D counter is reset upon timer D compare 4 event.

Bit 2 **CMP2**: Timer D compare 2 reset

The timer D counter is reset upon timer D compare 2 event.

Bit 1 **UPDT**: Timer D update reset

The timer D counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1

The timer D counter is reset upon timer F compare 1 event.

27.5.37 HRTIM timer E reset register (HRTIM_RSTER)

Address offset: 0x2D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
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
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2

The timer E counter is reset upon timer F Compare 2 event.

Bit 30 TIMD**CPM4**: Timer D compare 4

The timer E counter is reset upon timer D compare 4 event.

Bit 29 **T**IM**DCMP2**: Timer D compare 2

The timer E counter is reset upon timer D compare 2 event.

Bit 28 **T**IM**DCMP1**: Timer D compare 1

The timer E counter is reset upon timer D compare 1 event.

Bit 27 **T**IM**CCMP4**: Timer C compare 4

The timer E counter is reset upon timer C compare 4 event.

Bit 26 **T**IM**CCMP2**: Timer C compare 2

The timer E counter is reset upon timer C compare 2 event.

Bit 25 **T**IM**CCMP1**: Timer C compare 1

The timer E counter is reset upon timer C compare 1 event.

Bit 24 **T**IM**BCMP4**: Timer B compare 4

The timer E counter is reset upon timer B compare 4 event.

Bit 23 **T**IM**BCMP2**: Timer B compare 2

The timer E counter is reset upon timer B compare 2 event.

Bit 22 **T**IMB**CMP1**: Timer B compare 1

The timer E counter is reset upon timer B compare 1 event.

Bit 21 **T**IMA**CMP4**: Timer A compare 4

The timer E counter is reset upon timer A compare 4 event.

Bit 20 **T**IMA**CMP2**: Timer A compare 2

The timer E counter is reset upon timer A compare 2 event.

Bit 19 **T**IMA**CMP1**: Timer A compare 1

The timer E counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event

The timer E counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9

The timer E counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer E counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer E counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer E counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer E counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer E counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer E counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer E counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer E counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer E counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer E counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer E counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer E counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer E counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer E compare 4 reset
The timer E counter is reset upon timer E compare 4 event.

Bit 2 **CMP2**: Timer E compare 2 reset
The timer E counter is reset upon timer E compare 2 event.

Bit 1 **UPDT**: Timer E update reset
The timer E counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer E counter is reset upon timer F compare 1 event.

27.5.38 HRTIM timer F reset register (HRTIM_RSTFR)

Address offset: 0x354

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME CMP2	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
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**
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIME CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMECPM2**: Timer E compare 2
The timer F counter is reset upon timer E Compare 2 event.

Bit 30 TIMDC**PM4**: Timer D compare 4
The timer F counter is reset upon timer D compare 4 event.

Bit 29 **TIMDCMP2**: Timer D compare 2
The timer F counter is reset upon timer D compare 2 event.

Bit 28 **TIM**D**CMP1**: Timer D compare 1
The timer F counter is reset upon timer D compare 1 event.

Bit 27 **TIMCCMP4**: Timer C compare 4
The timer F counter is reset upon timer C compare 4 event.

Bit 26 **TIM**C**CMP2**: Timer C compare 2
The timer F counter is reset upon timer C compare 2 event.

Bit 25 **TIM**C**CMP1**: Timer C compare 1
The timer F counter is reset upon timer C compare 1 event.

Bit 24 **TIM**B**CMP4**: Timer B compare 4
The timer F counter is reset upon timer B compare 4 event.

Bit 23 **TIM**B**CMP2**: Timer B compare 2
The timer F counter is reset upon timer B compare 2 event.

Bit 22 **TIM**B**CMP1**: Timer B compare 1
The timer F counter is reset upon timer B compare 1 event.

Bit 21 **TIM**A**CMP4**: Timer A compare 4
The timer F counter is reset upon timer A compare 4 event.

Bit 20 **TIM**A**CMP2**: Timer A compare 2
The timer F counter is reset upon timer A compare 2 event.

Bit 19 **TIM**A**CMP1**: Timer A compare 1
The timer F counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event
The timer F counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9
The timer F counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer F counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer F counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer F counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer F counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer F counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer F counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer F counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer F counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer F counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer F counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer F counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer F counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer F counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer F compare 4 reset
The timer F counter is reset upon timer F compare 4 event.

Bit 2 **CMP2**: Timer F compare 2 reset
The timer F counter is reset upon timer F compare 2 event.

Bit 1 **UPDT**: Timer F update reset
The timer F counter is reset upon update event.

Bit 0 **TIMECMP1**: Timer E compare 1
The timer F counter is reset upon timer E compare 1 event.

27.5.39 HRTIM timer x chopper register (HRTIM_CHPxR) (x = A to F)

Address offset: Block A: 0x0D8

Address offset: Block B: 0x158

Address offset: Block C: 0x1D8

Address offset: Block D: 0x258

Address offset: Block E: 0x2D8

Address offset: Block F: 0x358

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.	STRTPW[3:0]				CARDTY[2:0)			CARFRQ[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:7 **STRPW[3:0]**: Timer x start pulsewidth

This register defines the initial pulsewidth following a rising edge on output signal.
This bitfield cannot be modified when one of the CHPx bits is set.
t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.

0000: 94.1 ns (1/10.625 MHz)
...
1111: 1.51 µs (16/10.625 MHz)

Bits 6:4 **CARDTY[2:0]**: Timer x chopper duty cycle value

This register defines the duty cycle of the carrier signal. This bitfield cannot be modified when one of the CHPx bits is set.
000: 0/8 (i.e. only 1st pulse is present)
...
111: 7/8

Bits 3:0 **CARFRQ[3:0]**: Timer x carrier frequency value

This register defines the carrier frequency F_{CHPFRQ} = f_{HRTIM} / (16 x (CARFRQ[3:0]+1)).
This bitfield cannot be modified when one of the CHPx bits is set.
0000: 10.625 MHz (f_{HRTIM}/ 16)
...
1111: 664.1 kHz (f_{HRTIM} / 256)

27.5.40 HRTIM timer A capture 1 control register (HRTIM_CPT1ACR)

Address offset: 0x0DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
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**
TF CMP2	TF CMP1	TF1 RST	TF1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

27.5.41 HRTIM timer B capture 1 control register (HRTIM_CPT1BCR)

Address offset: 0x15C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TF CMP2	TF CMP1	TF1 RST	TF1 SET
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 CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

27.5.42 HRTIM timer C capture 1 control register (HRTIM_CPT1CCR)

Address offset: 0x1DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TF CMP2	TF CMP1	TF1 RST	TF1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
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 CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

27.5.43 HRTIM timer D capture 1 control register (HRTIM_CPT1DCR)

Address offset: 0x25C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TF CMP2	TF CMP1	TF1 RST	TF1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
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 CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

27.5.44 HRTIM timer E capture 1 control register (HRTIM_CPT1ECR)

Address offset: 0x2DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TF CMP2	TF CMP1	TF1 RST	TF1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
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 CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

27.5.45 HRTIM timer F capture 1 control register (HRTIM_CPT1FCR)

Address offset: 0x35C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
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 CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

27.5.46 HRTIM timer x capture 2 control register (HRTIM_CPT2xCR) (x = A to F)

Address offset: Block A: 0x0E0

Address offset: Block B: 0x160

Address offset: Block C: 0x1E0

Address offset: Block D: 0x260

Address offset: Block E: 0x2E0

Address offset: Block F: 0x360

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TF CMP2 / TE CMP2	TF CMP1 / TE CMP1	TF1 RST / TE1 RST	TF1 SET / TE1 SET	TF CMP2 / TD CMP2	TF CMP1 / TD CMP1	TF1 RST / TD1 RST	TF1 SET / TD1 SET	TF CMP2 / TC CMP2	TF CMP1 / TC CMP1	TF1 RST / TC1 RST	TF1 SET / TC1 SET	TF CMP2 / TB CMP2	TF CMP1 / TB CMP1	TF1 RST / TB1 RST	TF1 SET / TB1 SET
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
TF CMP2 / TA CMP2	TF CMP1 / TA CMP1	TF1 RST / TA1 RST	TF1 SET / TA1 SET	EXEV1 0CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2FCR:

TECMP2: Timer E compare 2

Refer to TACMP2 description.

In HRTIM_CPT2ECR:

TFCMP2: Timer F compare 2

Refer to TACMP2 description.

Bit 30 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2FCR:

TECMP1: Timer E compare 1

Refer to TACMP1 description.

In HRTIM_CPT2ECR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 29 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2FCR:

TE1RST: Timer E output 1 reset

Refer to TA1RST description.

In HRTIM_CPT2ECR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 28 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2FCR:

TE1SET: Timer E output 1 set

Refer to TA1SET description.

In HRTIM_CPT2ECR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 27 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TDCMP2: Timer D compare 2

Refer to TACMP2 description.

In HRTIM_CPT2DCR:

TFCMP2: Timer F compare 2

Refer to TACMP2 description.

Bit 26 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TDCMP1:Timer D compare 1

Refer to TACMP1 description.

In HRTIM_CPT2DCR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 25 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TD1RST: Timer D output 1 reset

Refer to TA1RST description.

In HRTIM_CPT2DCR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 24 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TD1SET: Timer D output 1 set

Refer to TA1SET description.

In HRTIM_CPT2DCR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 23 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TCCMP2: Timer C compare 2

Refer to TACMP2 description.

In HRTIM_CPT2CCR:

TFCMP2: Timer F compare 2

Refer to TACMP2 description.

Bit 22 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TCCMP1:Timer C compare 1

Refer to TACMP1 description.

In HRTIM_CPT2CCR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 21 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TC1RST: Timer C output 1 reset

Refer to TA1RST description.

In HRTIM_CPT2CCR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 20 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TC1SET: Timer C output 1 set

Refer to TA1SET description.

In HRTIM_CPT2CCR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 19 In HRTIM_CPT2ACR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TBCMP2: Timer B compare 2

Refer to TACMP2 description.

In HRTIM_CPT2BCR:

TFCMP2: Timer F compare 2

Refer to TACMP2 description.

Bit 18 In HRTIM_CPT2ACR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TBCMP1: Timer B compare 1

Refer to TACMP1 description.

In HRTIM_CPT2BCR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 17 In HRTIM_CPT2ACR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TB1RST: Timer B output 1 reset

Refer to TA1RST description.

In HRTIM_CPT2BCR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 16 In HRTIM_CPT2ACR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TB1SET: Timer B output 1 set

Refer to TA1SET description.

In HRTIM_CPT2BCR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 15 In HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TACMP2: Timer A compare 2

0: No action

1: Timer A compare 2 triggers capture 2

In HRTIM_CPT2ACR:

TFCMP2: Timer E compare 2

Refer to TACMP2 description.

Bit 14 In HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TACMP1: Timer A compare 1

0: No action

1: Timer A compare 1 triggers capture 2

In HRTIM_CPT2ACR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 13 In HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TA1RST: Timer B output 1 reset

0: No action

1: Capture 2 is triggered by TA1 output active to inactive transition

In HRTIM_CPT2ACR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 12 In HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TA1SET: Timer B output 1 set

0: No action

1: Capture 2 is triggered by TA1 output inactive to active transition

In HRTIM_CPT2ACR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 11 **EXEV10CPT**: External event 10 capture

Refer to EXEV1CPT description

Bit 10 **EXEV9CPT**: External event 9 capture

Refer to EXEV1CPT description.

Bit 9 **EXEV8CPT**: External event 8 capture

Refer to EXEV1CPT description.

Bit 8 **EXEV7CPT**: External event 7 capture

Refer to EXEV1CPT description.

Bit 7 **EXEV6CPT**: External event 6 capture

Refer to EXEV1CPT description.

Bit 6 **EXEV5CPT**: External event 5 capture
Refer to EXEV1CPT description.

Bit 5 **EXEV4CPT**: External event 4 capture
Refer to EXEV1CPT description.

Bit 4 **EXEV3CPT**: External event 3 capture
Refer to EXEV1CPT description.

Bit 3 **EXEV2CPT**: External event 2 capture
Refer to EXEV1CPT description.

Bit 2 **EXEV1CPT**: External event 1 capture
0: No action
1: The external event 1 triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

27.5.47 HRTIM timer x output register (HRTIM_OUTxR) (x = A to F)

Address offset: Block A: 0x0E4

Address offset: Block B: 0x164

Address offset: Block C: 0x1E4

Address offset: Block D: 0x264

Address offset: Block E: 0x2E4

Address offset: Block F: 0x364

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.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM 2	POL2	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.	BIAR	Res.	DLYPRT[2:0]			DLYPR TEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM 1	POL1	Res.
	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 **DIDL2**: Output 2 deadtime upon burst mode Idle entry

This bit delays the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering in idle state during a burst mode operation.

0: The programmed Idle state is applied immediately to the output 2

1: Deadtime (inactive level) is inserted on output 2 before entering the idle mode. The deadtime value is set by DTFx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 is set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 22 **CHP2**: Output 2 chopper enable

This bit enables the chopper on output 2.

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 21:20 **FAULT2[1:0]**: Output 2 fault state

These bits select the output 2 state after a fault event.

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 19 **IDLES2**: Output 2 idle state

This bit selects the output 2 idle state.

0: Inactive

1: Active

Note: This parameter must be set prior to have the HRTIM controlling the outputs.

Bit 18 **IDLEM2**: Output 2 idle mode

This bit selects the output 2 idle mode.

0: No action: the output is not affected by the burst mode operation.

1: The output is in idle state when requested by the burst mode controller.

Note: This bit is preloaded and is changed during run-time, but must not be changed while the burst mode is active.

Bit 17 **POL2**: Output 2 polarity

This bit selects the output 2 polarity.

0: Positive polarity (output active high)

1: Negative polarity (output active low)

Note: This parameter cannot be changed once the timer x is enabled.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **BIAR**: Balanced Idle Automatic Resume

This bit selects is the outputs are automatically re-enabled after a balanced idle event.

This bit is only significant if DLYPRT[2:0] = 011 or 111, it is ignored otherwise.

0: Disabled

1: Enabled

Note: This parameter cannot be changed once the timer x is enabled.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DLYPRT[2:0]**: Delayed protection

These bits define the source and outputs on which the delayed protection schemes are applied.

In HRTIM_OUTAR, HRTIM_OUTBR, HRTIM_OUTCR:

000: Output 1 delayed idle on external event 6

001: Output 2 delayed idle on external event 6

010: Output 1 and output 2 delayed idle on external event 6

011: Balanced idle on external event 6

100: Output 1 delayed idle on external event 7

101: Output 2 delayed idle on external event 7

110: Output 1 and output 2 delayed idle on external event 7

111: Balanced idle on external event 7

In HRTIM_OUTDR, HRTIM_OUTER, HRTIM_OUTFR:

000: Output 1 delayed idle on external event 8

001: Output 2 delayed idle on external event 8

010: Output 1 and output 2 delayed idle on external event 8

011: Balanced idle on external event 8

100: Output 1 delayed idle on external event 9

101: Output 2 delayed idle on external event 9

110: Output 1 and output 2 delayed idle on external event 9

111: Balanced idle on external event 9

Note: This bitfield must not be modified once the delayed protection is enabled (DLYPRTEN bit set).

Bit 9 **DLYPRTEN**: Delayed protection enable

This bit enables the delayed protection scheme.

0: No action

1: Delayed protection is enabled, as per DLYPRT[2:0] bits

Note: This parameter cannot be changed once the timer x is enabled (TxEN bit set).

Bit 8 **DTEN**: Deadtime enable

This bit enables the deadtime insertion on output 1 and output 2.

0: Output 1 and output 2 signals are independent.

1: Deadtime is inserted between output 1 and output 2 (reference signal is output 1 signal generator)

Note: This parameter cannot be changed once the timer is operating (TxEN bit set) or if its outputs are enabled and set/reset by another timer.

Bit 7 **DIDL1**: Output 1 deadtime upon burst mode idle entry

This bit delays the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering the idle state during a burst mode operation.

0: The programmed idle state is applied immediately to the output 1

1: Deadtime (inactive level) is inserted on output 1 before entering the idle mode. The deadtime value is set by DTRx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 is set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 6 **CHP1**: Output 1 chopper enable

This bit enables the chopper on output 1.

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 5:4 **FAULT1[1:0]**: Output 1 fault state

These bits select the output 1 state after a fault event

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 3 **IDLES1**: Output 1 Idle State

This bit selects the output 1 idle state.

0: Inactive

1: Active

Note: This parameter must be set prior to HRTIM controlling the outputs.

Bit 2 **IDLEM1**: Output 1 Idle mode

This bit selects the output 1 idle mode.

0: No action: the output is not affected by the burst mode operation.

1: The output is in idle state when requested by the burst mode controller.

Note: This bit is preloaded and is changed during runtime, but must not be changed while burst mode is active.

Bit 1 **POL1**: Output 1 polarity

This bit selects the output 1 polarity.

0: Positive polarity (output active high)

1: Negative polarity (output active low)

Note: This parameter cannot be changed once the timer x is enabled.

Bit 0 Reserved, must be kept at reset value.

27.5.48 HRTIM timer x fault register (HRTIM_FLTxR) (x = A to F)

Address offset: Block A: 0x0E8

Address offset: Block B: 0x168

Address offset: Block C: 0x1E8

Address offset: Block D: 0x268

Address offset: Block E: 0x2E8

Address offset: Block F: 0x368

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT LCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rwo															
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.	FLT6 EN	FLT5 EN	FLT4 EN	FLT3 EN	FLT2 EN	FLT1 EN
										rw	rw	rw	rw	rw	rw

Bit 31 **FLTLCK**: Fault sources lock
0: FLT1EN..FLT6EN bits are read/write
1: FLT1EN..FLT6EN bits are read only
The FLTLCK bit is write-once. Once it has been set, it cannot be modified till the next system reset.

Bits 30:6 Reserved, must be kept at reset value.

Bit 5 **FLT6EN**: Fault 6 enable
0: Fault 6 input ignored
1: Fault 6 input is active and disables HRTIM outputs

Bit 4 **FLT5EN**: Fault 5 enable
0: Fault 5 input ignored
1: Fault 5 input is active and disables HRTIM outputs

Bit 3 **FLT4EN**: Fault 4 enable
0: Fault 4 input ignored
1: Fault 4 input is active and disables HRTIM outputs

Bit 2 **FLT3EN**: Fault 3 enable
0: Fault 3 input ignored
1: Fault 3 input is active and disables HRTIM outputs

Bit 1 **FLT2EN**: Fault 2 enable
0: Fault 2 input ignored
1: Fault 2 input is active and disables HRTIM outputs

Bit 0 **FLT1EN**: Fault 1 enable
0: Fault 1 input ignored
1: Fault 1 input is active and disables HRTIM outputs

27.5.49 HRTIM timer x control register 2 (HRTIM_TIMxCR2) (x = A to F)

Address offset: Block A: 0x0EC

Address offset: Block B: 0x16C

Address offset: Block C: 0x1EC

Address offset: Block D: 0x26C

Address offset: Block E: 0x2EC

Address offset: Block F: 0x36C

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.	TRG HLF	Res.	Res.	GT CMP3	GT CMP1
											rw			rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FEROM[1:0]		BMROM[1:0]		ADROM[1:0]		OUTROM[1:0]		ROM[1:0]		Res.	UDM	Res.	DCDR	DCDS	DCDE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 20 **TRGHLF**: Triggered-half mode

This bitfield defines whether the compare 2 register is behaving in standard mode (compare match issued as soon as counter equal compare), or in triggered-half mode (see *Section : Triggered-half mode*).

0: CMP2 register is written by the user only (standard compare mode)

1: CMP2 value is set by hardware as soon as a capture 1 event occurs. It is loaded with the (capture 1 divided by 2) value. The initial value can be written by the user (as long as TRGHLF is reset), but is ignored once the first capture has been triggered (the preload mechanism is disabled for CMP2 when the TRGHLF bit is set).

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **GTCMP3**: Greater than compare 3 PWM mode

This bit defines the compare 3 operating mode.

0: The compare 3 event is generated when the counter is equal to the compare value (compare match mode)

1: The compare 3 event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

Bit 16 **GTCMP1**: Greater than compare 1 PWM mode

This bit defines the compare 1 operating mode.

0: The compare 1 event is generated when the counter is equal to the compare value (compare match mode)

1: The compare 1 event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

Bits 15:14 **FEROM[1:0]**: Fault and event roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the Roll-over event used by the fault and event counters.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value
01: Event generated when the counter is equal to 0
10: Event generated when the counter is equal to HRTIM_PERxR
11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bits 13:12 **BMROM[1:0]**: Burst mode roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event used in the burst mode controller, as clock as burst mode trigger.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value
01: Event generated when the counter is equal to 0
10: Event generated when the counter is equal to HRTIM_PERxR
11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This parameter cannot be changed once the timer is operating (TxEN bit set).

Bits 11:10 **ADROM[1:0]**: ADC roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event which triggers the ADC.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value
01: Event generated when the counter is equal to 0
10: Event generated when the counter is equal to HRTIM_PERxR
11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bits 9:8 **OUTROM[1:0]**: Output roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event which sets and/or resets the ouputs, as per HRTIM_SETxyR and HRTIM_RSTxyR settings.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value
01: Event generated when the counter is equal to 0
10: Event generated when the counter is equal to HRTIM_PERxR
11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bits 7:6 **ROM[1:0]**: Roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event with the following destinations: update trigger (to transfer content from preload to active registers), IRQ and DMA requests, repetition counter decrement and external event filtering.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value
01: Event generated when the counter is equal to 0
10: Event generated when the counter is equal to HRTIM_PERxR
11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bit 5 Reserved, must be kept at reset value.

Bit 4 **UDM**: Up-Down mode

This bit defines if the counter is operating in up or up-down counting mode.

0: The counter is operating in up-counting mode

1: The counter is operating in up-down counting mode

Note: This bit cannot be changed once the timer is operating (TxEN bit set).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCDR**: Dual channel DAC reset trigger

This bit defines when the hrtim_dac_reset_trgx trigger is generated.

0: The trigger is generated on counter reset or roll-over event

1: The trigger is generated on output 1 set event

Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

Bit 1 **DCDS** Dual channel DAC Step trigger

This bit defines when the hrtim_dac_step_trgx trigger is generated.

0: The trigger is generated on compare 2 event

1: The trigger is generated on output 1 reset event

Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

Bit 0 **DCDE**: Dual channel DAC trigger enable

This bit enables the dual channel DAC triggering mechanism.

0: Dual channel DAC trigger disabled

1: Dual channel DAC trigger enabled

Note: This bit cannot be changed once the timer is operating (TxEN bit set).

27.5.50 HRTIM timer x external event filtering register 3 (HRTIM_TIMxEEFR3) (x = A to F)

Address offset: Block A: 0x0F0

Address offset: Block B: 0x170

Address offset: Block C: 0x1F0

Address offset: Block D: 0x270

Address offset: Block E: 0x2F0

Address offset: Block F: 0x370

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.	EEVACNT[5:0]						EEVASEL[3:0]				Res.	EEVA RSTM	EEVA CRES	EEVA CE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **EEVACNT[5:0]** External event A counter

This bitfield selects the external event A counter threshold. An event is considered valid when the number of events is equal to the (EEVACNT[5:0]+1) value.

Bits 7:4 **EEVASEL[3:0]**: External event A Selection

This bit selects the external event A source.

0: External event 1 is used as external event A source

1: External event 2 is used as external event A source

...

9: External event 10 is used as external event A source

Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EEVARSTM**: External event A reset mode

This bit selects the external event x counter reset mode.

0: External event counter A is reset on each reset / roll-over event

1: External event counter A is reset on each reset / roll-over event only if no event occurs during last counting period

Bit 1 **EEVACRES**: External event A counter reset

This bit resets the external event A counter. It is set by software and reset by hardware.

0: No action

1: External event counter A is reset

Bit 0 **EEVACE**: External event A counter enable

This bit enables the external event x counter.

0: External event A counter disabled

1: External event A counter enabled

27.5.51 HRTIM control register 1 (HRTIM_CR1)

Address offset: 0x380

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]		
				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.	Res.	Res.	Res.	Res.	Res.	TF UDIS	TE UDIS	TD UDIS	TC UDIS	TB UDIS	TA UDIS	MUDIS
									rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:25 **AD4USRC[2:0]**: ADC trigger 4 update source

Refer to AD1USRC[2:0] description.

Bits 24:22 **AD3USRC[2:0]**: ADC trigger 3 update source

Refer to AD1USRC[2:0] description.

Bits 21:19 **AD2USRC[2:0]**: ADC trigger 2 update source

Refer to AD1USRC[2:0] description.

Bits 18:16 **AD1USRC[2:0]**: ADC trigger 1 update source

These bits define the source which triggers the update of the HRTIM_ADC1R register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, in HRTIM_MCR or HRTIM_TIMxCR.

000: Master timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110: Timer F
111: Reserved

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

Bit 6 **TFUDIS**: Timer F update disable

Refer to TAUDIS description.

Bit 5 **TEUDIS**: Timer E update disable

Refer to TAUDIS description

Bit 4 **TDUDIS**: Timer D update disable

Refer to TAUDIS description.

Bit 3 **TCUDIS**: Timer C update disable

Refer to TAUDIS description.

Bit 2 **TBUDIS**: Timer B update disable

Refer to TAUDIS description.

Bit 1 **TAUDIS**: Timer A update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily on timer A.

0: Update enabled. The update occurs upon generation of the selected source.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 0 **MUDIS**: Master update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily.

0: Update enabled.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

27.5.52 HRTIM control register 2 (HRTIM_CR2)

Address offset: 0x384

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.	SWPF	SWPE	SWPD	SWPC	SWPB	SWPA
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TFRST	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	TF SWU	TE SWU	TD SWU	TC SWU	TB SWU	TA SWU	MSWU
	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 **SWPF**: Swap timer F outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 20 **SWPE**: Swap timer E outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 19 **SWPD**: Swap timer D outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 18 **SWPC**: Swap timer C outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 17 **SWPB**: Swap timer B outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 16 **SWPA**: Swap timer A outputs

This bit allows to swap the timer A outputs.

0: HRTIM_SETA1R and HRTIM_RSTA1R are coding for the output A1, HRTIM_SETA2R and HRTIM_RSTA2R are coding for the output A2

1: HRTIM_SETA1R and HRTIM_RSTA1R are coding for the output A2, HRTIM_SETA2R and HRTIM_RSTA2R are coding for the output A1

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TFRST**: Timer F counter software reset

Refer to TARST description.

Bit 13 **TERST**: Timer E counter software reset

Refer to TARST description.

Bit 12 **TDRST**: Timer D counter software reset

Refer to TARST description.

Bit 11 **TCRST**: Timer C counter software reset

Refer to TARST description.

Bit 10 **TBRST**: Timer B counter software reset
Refer to TARST description.

Bit 9 **TARST**: Timer A counter software reset
Setting this bit resets the timer A counter.
The bit is automatically reset by hardware.

Bit 8 **MRST**: Master counter software reset
Setting this bit resets the master timer counter.
The bit is automatically reset by hardware.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TFSWU**: Timer F software update
Refer to TASWU description.

Bit 5 **TESWU**: Timer E software update
Refer to TASWU description.

Bit 4 **TDSWU**: Timer D software update
Refer to TASWU description.

Bit 3 **TCSWU**: Timer C software update
Refer to TASWU description.

Bit 2 **TBSWU**: Timer B software update
Refer to TASWU description.

Bit 1 **TASWU**: Timer A software update
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register and any pending update request is canceled.

Bit 0 **MSWU**: Master timer software update
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register in the master timer and any pending update request is canceled.

27.5.53 HRTIM interrupt status register (HRTIM_ISR)

Address offset: 0x388

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.	BMPER	DLLRDY
														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.	FLT6	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
									r		r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPER**: Burst mode period interrupt flag

This bit is set by hardware when a single-shot burst mode operation is completed or at the end of a burst mode period in continuous mode. It is cleared by software writing it at 1.

0: No burst mode period interrupt occurred

1: Burst mode period interrupt occurred

Bit 16 **DLLRDY**: DLL Ready Interrupt Flag

This bit is set by hardware when the DLL calibration is completed. It is cleared by software writing it at 1.

0: No DLL calibration ready interrupt occurred

1: DLL calibration ready interrupt occurred

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

Bit 6 **FLT6**: Fault 6 interrupt flag

Refer to FLT1 description.

Bit 5 **SYSFLT**: System fault interrupt flag

Refer to FLT1 description.

Bit 4 **FLT5**: Fault 5 interrupt flag

Refer to FLT1 description.

Bit 3 **FLT4**: Fault 4 interrupt flag

Refer to FLT1 description.

Bit 2 **FLT3**: Fault 3 interrupt flag

Refer to FLT1 description.

Bit 1 **FLT2**: Fault 2 interrupt flag

Refer to FLT1 description.

Bit 0 **FLT1**: Fault 1 interrupt flag

This bit is set by hardware when fault 1 event occurs. It is cleared by software writing it at 1.

0: No fault 1 interrupt occurred

1: Fault 1 interrupt occurred

27.5.54 HRTIM interrupt clear register (HRTIM_ICR)

Address offset: 0x38C

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.	BMPERC	DLLRDYC
														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.	FLT6C	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
									w	w	w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPERC**: Burst mode period flag clear

Writing 1 to this bit clears the BMPER flag in HRTIM_ISR register.

Bit 16 **DLLRDYC**: DLL Ready Interrupt flag Clear

Writing 1 to this bit clears the DLLRDY flag in HRTIM_ISR register.

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

Bit 6 **FLT6C**: Fault 6 interrupt flag clear

Writing 1 to this bit clears the FLT6 flag in HRTIM_ISR register.

Bit 5 **SYSFLTC**: System fault interrupt flag clear

Writing 1 to this bit clears the SYSFLT flag in HRTIM_ISR register.

Bit 4 **FLT5C**: Fault 5 interrupt flag clear

Writing 1 to this bit clears the FLT5 flag in HRTIM_ISR register.

Bit 3 **FLT4C**: Fault 4 interrupt flag clear

Writing 1 to this bit clears the FLT4 flag in HRTIM_ISR register.

Bit 2 **FLT3C**: Fault 3 interrupt flag clear

Writing 1 to this bit clears the FLT3 flag in HRTIM_ISR register.

Bit 1 **FLT2C**: Fault 2 interrupt flag clear

Writing 1 to this bit clears the FLT2 flag in HRTIM_ISR register.

Bit 0 **FLT1C**: Fault 1 interrupt flag clear

Writing 1 to this bit clears the FLT1 flag in HRTIM_ISR register.

27.5.55 HRTIM interrupt enable register (HRTIM_IER)

Address offset: 0x390

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.	BMPE RIE	DLL RDYIE
														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.	FLT6 IE	SYSFL TIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPERIE**: Burst mode period interrupt enable

This bit is set and cleared by software to enable/disable the burst mode period interrupt.
0: Burst mode period interrupt disabled
1: Burst mode period interrupt enabled

Bit 16 **DLLRDYIE**: DLL Ready Interrupt Enable

This bit is set and cleared by software to enable/disable the DLL ready interrupt.
0: DLL ready interrupt disabled
1: DLL ready interrupt enabled

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

Bit 6 **FLT6IE**: Fault 6 interrupt enable

Refer to FLT1IE description.

Bit 5 **SYSFLTIE**: System fault interrupt enable

Refer to FLT1IE description.

Bit 4 **FLT5IE**: Fault 5 interrupt enable

Refer to FLT1IE description.

Bit 3 **FLT4IE**: Fault 4 interrupt enable

Refer to FLT1IE description.

Bit 2 **FLT3IE**: Fault 3 interrupt enable

Refer to FLT1IE description.

Bit 1 **FLT2IE**: Fault 2 interrupt enable

Refer to FLT1IE description.

Bit 0 **FLT1IE**: Fault 1 interrupt enable

This bit is set and cleared by software to enable/disable the fault 1 interrupt.
0: Fault 1 interrupt disabled
1: Fault 1 interrupt enabled

27.5.56 HRTIM output enable register (HRTIM_OENR)

Address offset: 0x394

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.	TF2O EN	TF1O EN	TE2O EN	TE1O EN	TD2O EN	TD1O EN	TC2O EN	TC1O EN	TB2O EN	TB1O EN	TA2O EN	TA1O EN
				rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2OEN**: Timer F output 2 enable
Refer to TA1OEN description.

Bit 10 **TF1OEN**: Timer F output 1 enable
Refer to TA1OEN description

Bit 9 **TE2OEN**: Timer E output 2 enable
Refer to TA1OEN description.

Bit 8 **TE1OEN**: Timer E output 1 enable
Refer to TA1OEN description.

Bit 7 **TD2OEN**: Timer D output 2 enable
Refer to TA1OEN description.

Bit 6 **TD1OEN**: Timer D output 1 enable
Refer to TA1OEN description.

Bit 5 **TC2OEN**: Timer C output 2 enable
Refer to TA1OEN description.

Bit 4 **TC1OEN**: Timer C output 1 enable
Refer to TA1OEN description.

Bit 3 **TB2OEN**: Timer B output 2 enable
Refer to TA1OEN description.

Bit 2 **TB1OEN**: Timer B output 1 enable
Refer to TA1OEN description.

Bit 1 **TA2OEN**: Timer A output 2 enable
Refer to TA1OEN description.

Bit 0 **TA1OEN**: Timer A output 1 enable
Setting this bit enables the timer A output 1. Writing “0” has no effect.
Reading the bit returns the output enable/disable status.
This bit is cleared asynchronously by hardware as soon as the timer-related fault input(s) is (are) active.
0: output A1 disabled. The output is either in fault or Idle state.
1: output A1 enabled

Note: The disable status corresponds to both idle and fault states. The output disable status is given by TA1ODS bit in the HRTIM_ODSR register.

27.5.57 HRTIM output disable register (HRTIM_ODISR)

Address offset: 0x398

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.	TF2ODIS	TF1ODIS	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
				w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2ODIS**: Timer F output 2 disable

Refer to TA1ODIS description.

Bit 10 **TF1ODIS**: Timer F output 1 disable

Refer to TA1ODIS description

Bit 9 **TE2ODIS**: Timer E output 2 disable

Refer to TA1ODIS description.

Bit 8 **TE1ODIS**: Timer E output 1 disable

Refer to TA1ODIS description.

Bit 7 **TD2ODIS**: Timer D output 2 disable

Refer to TA1ODIS description.

Bit 6 **TD1ODIS**: Timer D output 1 disable

Refer to TA1ODIS description.

Bit 5 **TC2ODIS**: Timer C output 2 disable

Refer to TA1ODIS description.

Bit 4 **TC1ODIS**: Timer C output 1 disable

Refer to TA1ODIS description.

Bit 3 **TB2ODIS**: Timer B output 2 disable

Refer to TA1ODIS description.

Bit 2 **TB1ODIS**: Timer B output 1 disable

Refer to TA1ODIS description.

Bit 1 **TA2ODIS**: Timer A output 2 disable

Refer to TA1ODIS description.

Bit 0 **TA1ODIS**: Timer A output 1 disable

Setting this bit disables the timer A output 1. The output enters the idle state, either from the run state or from the fault state.

Writing “0” has no effect.

27.5.58 HRTIM output disable status register (HRTIM_ODSR)

Address offset: 0x39C

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.	TF2 ODS	TF1 ODS	TE2 ODS	TE1 ODS	TD2 ODS	TD1 ODS	TC2 ODS	TC1 ODS	TB2 ODS	TB1 ODS	TA2 ODS	TA1 ODS
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2ODS**: Timer F output 2 disable status
Refer to TA1ODS description.

Bit 10 **TF1ODS**: Timer F output 1 disable status
Refer to TA1ODS description.

Bit 9 **TE2ODS**: Timer E output 2 disable status
Refer to TA1ODS description.

Bit 8 **TE1ODS**: Timer E output 1 disable status
Refer to TA1ODS description.

Bit 7 **TD2ODS**: Timer D output 2 disable status
Refer to TA1ODS description.

Bit 6 **TD1ODS**: Timer D output 1 disable status
Refer to TA1ODS description.

Bit 5 **TC2ODS**: Timer C output 2 disable status
Refer to TA1ODS description.

Bit 4 **TC1ODS**: Timer C output 1 disable status
Refer to TA1ODS description.

Bit 3 **TB2ODS**: Timer B output 2 disable status
Refer to TA1ODS description.

Bit 2 **TB1ODS**: Timer B output 1 disable status
Refer to TA1ODS description.

Bit 1 **TA2ODS**: Timer A output 2 disable status
Refer to TA1ODS description.

Bit 0 **TA1ODS**: Timer A output 1 disable status
Reading the bit returns the output disable status. It is not significant when the output is active (Tx1OEN or Tx2OEN = 1).
0: Output A1 disabled, in Idle state.
1: Output A1 disabled, in fault state.

27.5.59 HRTIM burst mode control register (HRTIM_BMCR)

Address offset: 0x3A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFBM	TEBM	TDBM	TCBM	TBBM	TABM	MTBM
rc_w0									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.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BMSTAT**: Burst mode status

This bit gives the current operating state.

0: Normal operation

1: Burst operation on-going. Writing this bit to 0 causes a burst mode early termination.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **TFBM**: Timer F burst mode

Refer to TABM description.

Bit 21 **TEBM**: Timer E burst mode

Refer to TABM description.

Bit 20 **TDBM**: Timer D burst mode

Refer to TABM description.

Bit 19 **TCBM**: Timer C burst mode

Refer to TABM description.

Bit 18 **TBBM**: Timer B burst mode

Refer to TABM description.

Bit 17 **TABM**: Timer A burst mode

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: TA counter clock is maintained and the timer operates normally

1: TA counter clock is stopped and the counter is reset

Note: This bit must not be set when the balanced idle mode is active (DLYPRT[2:0] = 0x11).

Bit 16 **MTBM**: Master timer burst mode

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: Master Timer counter clock is maintained and the timer operates normally

1: Master Timer counter clock is stopped and the counter is reset

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **BMPREN**: Burst mode preload enable

This bit enables the registers preload mechanism and defines whether a write access into a preloadable register (HRTIM_BMCMPR, HRTIM_BMPER) is done into the active or the preload register.

0: Preload disabled: the write access is directly done into active registers

1: Preload enabled: the write access is done into preload registers

Bits 9:6 **BMPRSC[3:0]**: Burst mode prescaler

Defines the prescaling ratio of the f_{HRTIM} clock for the burst mode controller. This bitfield cannot be changed while the burst mode is enabled.

0000: Clock not divided
0001: Division by 2
0010: Division by 4
0011: Division by 8
0100: Division by 16
0101: Division by 32
0110: Division by 64
0111: Division by 128
1000: Division by 256
1001: Division by 512
1010: Division by 1024
1011: Division by 2048
1100: Division by 4096
1101: Division by 8192
1110: Division by 16384
1111: Division by 32768

Bits 5:2 **BMCLK[3:0]**: Burst mode clock source

This bitfield defines the clock source for the burst mode counter. It cannot be changed while the burst mode is enabled (refer to *Table 210* for on-chip events 1..4 connections details).

0000: Master timer counter reset/roll-over
0001: Timer A counter reset/roll-over
0010: Timer B counter reset/roll-over
0011: Timer C counter reset/roll-over
0100: Timer D counter reset/roll-over
0101: Timer E counter reset/roll-over
0110: On-chip event 1 (hrtim_bm_ck1), acting as a burst mode counter clock
0111: On-chip event 2 (hrtim_bm_ck2) acting as a burst mode counter clock
1000: On-chip event 3 (hrtim_bm_ck3) acting as a burst mode counter clock
1001: On-chip event 4 (hrtim_bm_ck4) acting as a burst mode counter clock
1010: Prescaled f_{HRTIM} clock (as per BMPRSC[3:0] setting)
1011: Timer F counter reset/roll-over
Others: Reserved

Bit 1 **BMOM**: Burst mode operating mode

This bit defines if the burst mode is entered once or if it is continuously operating.

0: Single-shot mode
1: Continuous operation

Bit 0 **BME**: Burst mode enable

This bit starts the burst mode controller which becomes ready to receive the start trigger.
Writing this bit to 0 causes a burst mode early termination.

0: Burst mode disabled
1: Burst mode enabled

27.5.60 HRTIM burst mode trigger register (HRTIM_BMTRGR)

Address offset: 0x3A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OCHP EV	EEV8	EEV7	TDEEV 8	TAEEV 7	TECMP 2	TECMP 1	TEREP	TF CMP1	TDCM P2	TF REP	TDREP	TDRST	TF RST	TCCM P1	TCREP
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
TCRST	TBCMP 2	TBCMP 1	TBREP	TBRST	TACMP 2	TACMP 1	TAREP	TARST	MSTC MP4	MSTC MP3	MSTC MP2	MSTC MP1	MSTRE P	MSTRS T	SW
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OCHPEV**: On-chip event

A rising edge on the hrtim_bm_trg input (connected to general purpose timer TRGO output) triggers a burst mode entry (see *Table 207* for details).

Bit 30 **EEV8**: External event 8 (TIMD filters applied)

The external event 8 conditioned by TIMD filters is starting the burst mode operation.

Bit 29 **EEV7**: External event 7 (TIMA filters applied)

The external event 7 conditioned by TIMA filters is starting the burst mode operation.

Bit 28 **TDEEV8**: Timer D period following external event 8

The timer D period following an external event 8 (conditioned by TIMD filters) is starting the burst mode operation.

Bit 27 **TAEEV7**: Timer A period following external event 7

The timer A period following an external event 7 (conditioned by TIMA filters) is starting the burst mode operation.

Bit 26 **TECMP2**: Timer E compare 2 event

Refer to TACMP1 description.

Bit 25 **TECMP1**: Timer E compare 1 event

Refer to TACMP1 description.

Bit 24 **TEREP**: Timer E repetition

Refer to TAREP description.

Bit 23 **TFCMP1**: Timer F compare 1 event

Refer to TACMP1 description.

Bit 22 **TDCMP2**: Timer D compare 2 event

Refer to TACMP1 description.

Bit 21 **TFREP**: Timer F repetition

Refer to TAREP description.

Bit 20 **TDREP**: Timer D repetition

Refer to TAREP description.

Bit 19 **TDRST**: Timer D reset or roll-over

Refer to TARST description.

Bit 18 **TFRST**: Timer F reset

Refer to TARST description.

Bit 17 **TCCMP1**: Timer C compare 1 event
Refer to TACMP1 description.

Bit 16 **TCREP**: Timer C repetition
Refer to TAREP description.

Bit 15 **TCRST**: Timer C reset or roll-over
Refer to TARST description.

Bit 14 **TBCMP2**: Timer B compare 2 event
Refer to TACMP1 description.

Bit 13 **TBCMP1**: Timer B compare 1 event
Refer to TACMP1 description.

Bit 12 **TBREP**: Timer B repetition
Refer to TAREP description.

Bit 11 **TBRST**: Timer B reset or roll-over
Refer to TARST description.

Bit 10 **TACMP2**: Timer A compare 2 event
Refer to TACMP1 description.

Bit 9 **TACMP1**: Timer A compare 1 event
The timer A compare 1 event is starting the burst mode operation.

Bit 8 **TAREP**: Timer A repetition
The Timer A repetition event is starting the burst mode operation.

Bit 7 **TARST**: Timer A reset or roll-over
The Timer A reset or roll-over event is starting the burst mode operation.

Bit 6 **MSTCMP4**: Master compare 4
Refer to MSTCMP1 description.

Bit 5 **MSTCMP3**: Master compare 3
Refer to MSTCMP1 description.

Bit 4 **MSTCMP2**: Master compare 2
Refer to MSTCMP1 description.

Bit 3 **MSTCMP1**: Master compare 1
The master timer compare 1 event is starting the burst mode operation.

Bit 2 **MSTREP**: Master repetition
The master timer repetition event is starting the burst mode operation.

Bit 1 **MSTRST**: Master reset or roll-over
The master timer reset and roll-over event is starting the burst mode operation.

Bit 0 **SW**: Software start
This bit is set by software and automatically reset by hardware.
When set, It starts the burst mode operation immediately.
This bit is not active if the burst mode is not enabled (BME bit is reset).

27.5.61 HRTIM burst mode compare register (HRTIM_BMCMPR)

Address offset: 0x3A8

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
BMCMP[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 **BMCMP[15:0]**: Burst mode compare value

Defines the number of periods during which the selected timers are in idle state.

This register holds either the content of the preload register or the content of the active register if the preload is disabled.

Note: BMCMP[15:0] cannot be set to 0x0000 when using the f_{HRTIM} *clock without a prescaler as the burst mode clock source (BMCLK[3:0] = 1010 and BMPRESC[3:0] = 0000).*

27.5.62 HRTIM burst mode period register (HRTIM_BMPER)

Address offset: 0x3AC

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
BMPER[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 **BMPER[15:0]**: Burst mode period

Defines the burst mode repetition period.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

Note: The BMPER[15:0] must not be null when the burst mode is enabled.

27.5.63 HRTIM timer external event control register 1 (HRTIM_EECR1)

Address offset: 0x3B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1]
		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
EE3SNS[0]	EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[1:0]	
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 **EE5FAST**: External event 5 fast mode

Refer to EE1FAST description.

Bits 28:27 **EE5SNS[1:0]**: External event 5 sensitivity

Refer to EE1SNS[1:0] description.

Bit 26 **EE5POL**: External event 5 polarity

Refer to EE1POL description.

Bits 25:24 **EE5SRC[1:0]**: External event 5 source

Refer to EE1SRC[1:0] description.

Bit 23 **EE4FAST**: External event 4 fast mode

Refer to EE1FAST description.

Bits 22:21 **EE4SNS[1:0]**: External event 4 sensitivity

Refer to EE1SNS[1:0] description.

Bit 20 **EE4POL**: External event 4 polarity

Refer to EE1POL description.

Bits 19:18 **EE4SRC[1:0]**: External event 4 source

Refer to EE1SRC[1:0] description.

Bit 17 **EE3FAST**: External event 3 fast mode

Refer to EE1FAST description.

Bits 16:15 **EE3SNS[1:0]**: External event 3 sensitivity

Refer to EE1SNS[1:0] description.

Bit 14 **EE3POL**: External event 3 polarity

Refer to EE1POL description.

Bits 13:12 **EE3SRC[1:0]**: External event 3 source

Refer to EE1SRC[1:0] description.

Bit 11 **EE2FAST**: External event 2 fast mode

Refer to EE1FAST description.

Bits 10:9 **EE2SNS[1:0]**: External event 2 sensitivity

Refer to EE1SNS[1:0] description.

Bit 8 **EE2POL**: External event 2 polarity

Refer to EE1POL description.

Bits 7:6 **EE2SRC[1:0]**: External event 2 source

Refer to EE1SRC[1:0] description.

Bit 5 **EE1FAST**: External event 1 fast mode

0: External event 1 is re-synchronized by the HRTIM logic before acting on outputs, which adds a f_{HRTIM} clock-related latency
1: External event 1 is acting asynchronously on outputs (low latency mode)

Note: This bit must not be modified once the counter in which the event is used is enabled (TxCEN bit set).

Bits 4:3 **EE1SNS[1:0]**: External event 1 sensitivity

00: On active level defined by EE1POL bit
01: Rising edge, whatever EE1POL bit value
10: Falling edge, whatever EE1POL bit value
11: Both edges, whatever EE1POL bit value

Bit 2 **EE1POL**: External event 1 polarity

This bit is only significant if EE1SNS[1:0] = 00.
0: External event is active high
1: External event is active low

Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

Bits 1:0 **EE1SRC[1:0]**: External event 1 source

This bitfield selects the External event 1 source. See *Table 208* for details.
00: hrtim_eev1_1
01: hrtim_eev1_2
10: hrtim_eev1_3
11: hrtim_eev1_4

Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

27.5.64 HRTIM timer external event control register 2 (HRTIM_EECR2)

Address offset: 0x3B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1]
			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
EE8SNS[0]	EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[1:0]	
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:27 **EE10SNS[1:0]**: External event 10 sensitivity
Refer to EE1SNS[1:0] description.

Bit 26 **EE10POL**: External event 10 polarity
Refer to EE1POL description.

Bits 25:24 **EE10SRC[1:0]**: External event 10 source
Refer to EE1SRC[1:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EE9SNS[1:0]**: External event 9 sensitivity
Refer to EE1SNS[1:0] description.

Bit 20 **EE9POL**: External event 9 polarity
Refer to EE1POL description.

Bits 19:18 **EE9SRC[1:0]**: External event 9 source
Refer to EE1SRC[1:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:15 **EE8SNS[1:0]**: External event 8 sensitivity
Refer to EE1SNS[1:0] description.

Bit 14 **EE8POL**: External event 8 polarity
Refer to EE1POL description.

Bits 13:12 **EE8SRC[1:0]**: External event 8 source
Refer to EE1SRC[1:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EE7SNS[1:0]**: External event 7 sensitivity
Refer to EE1SNS[1:0] description.

Bit 8 **EE7POL**: External event 7 polarity
Refer to EE1POL description.

Bits 7:6 **EE7SRC[1:0]**: External event 7 source
Refer to EE1SRC[1:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EE6SNS[1:0]**: External event 6 sensitivity

Refer to EE1SNS[1:0] description.

Bit 2 **EE6POL**: External event 6 polarity

Refer to EE1POL description.

Bits 1:0 **EE6SRC[1:0]**: External event 6 source

Refer to EE1SRC[1:0] description.

27.5.65 HRTIM timer external event control register 3 (HRTIM_EECR3)

Address offset: 0x3B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EEVSD[1:0]		Res.	Res.	EE10F[3:0]				Res.	Res.	EE9F[3:0]				Res.	Res.
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
EE8F[3:0]				Res.	Res.	EE7F[3:0]				Res.	Res.	EE6F[3:0]			
rw	rw	rw	rw			rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:30 **EEVSD[1:0]**: External event sampling clock division

This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the external event signal sampling clock (f_{EEVS}) used by the digital filters.

00: f_{EEVS}=f_{HRTIM}
01: f_{EEVS}=f_{HRTIM} / 2
10: f_{EEVS}=f_{HRTIM} / 4
11: f_{EEVS}=f_{HRTIM} / 8

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

Bits 27:24 **EE10F[3:0]**: External event 10 filter

Refer to EE6F[3:0] description.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:18 **EE9F[3:0]**: External event 9 filter

Refer to EE6F[3:0] description.

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:12 **EE8F[3:0]**: External event 8 filter

Refer to EE6F[3:0] description.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **EE7F[3:0]**: External event 7 filter

Refer to EE6F[3:0] description.

Bits 4:5 Reserved, must be kept at reset value.

Bits 3:0 **EE6F[3:0]**: External event 6 filter

This bitfield defines the frequency used to sample external event 6 input and the length of the digital filter applied to EEV6. The digital filter is made of a counter in which N valid samples are needed to validate a transition on the output.

0000: Filter disabled
0001: $f_{SAMPLING}= f_{HRTIM}$, N=2
0010: $f_{SAMPLING}= f_{HRTIM}$, N=4
0011: $f_{SAMPLING}= f_{HRTIM}$, N=8
0100: $f_{SAMPLING}= f_{EEVS}/2$, N=6
0101: $f_{SAMPLING}= f_{EEVS}/2$, N=8
0110: $f_{SAMPLING}= f_{EEVS}/4$, N=6
0111: $f_{SAMPLING}= f_{EEVS}/4$, N=8
1000: $f_{SAMPLING}= f_{EEVS}/8$, N=6
1001: $f_{SAMPLING}= f_{EEVS}/8$, N=8
1010: $f_{SAMPLING}= f_{EEVS}/16$, N=5
1011: $f_{SAMPLING}= f_{EEVS}/16$, N=6
1100: $f_{SAMPLING}= f_{EEVS}/16$, N=8
1101: $f_{SAMPLING}= f_{EEVS}/32$, N=5
1110: $f_{SAMPLING}= f_{EEVS}/32$, N=6
1111: $f_{SAMPLING}= f_{EEVS}/32$, N=8

27.5.66 HRTIM ADC trigger 1 register (HRTIM_ADC1R)

Address offset: 0x3BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC1 TEPER	ADC1 TEC4	ADC1 TEC3	ADC1 TFRST	ADC1 TDPER	ADC1 TDC4	ADC1 TDC3	ADC1 TFPER	ADC1 TCPER	ADC1 TCC4	ADC1 TCC3	ADC1 TFC4	ADC1 TBRST	ADC1 TBPER	ADC1 TBC4	ADC1 TBC3
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
ADC1 TFC3	ADC1 TARST	ADC1 TAPER	ADC1 TAC4	ADC1 TAC3	ADC1 TFC2	ADC1 EEV5	ADC1 EEV4	ADC1 EEV3	ADC1 EEV2	ADC1 EEV1	ADC1 MPER	ADC1 MC4	ADC1 MC3	ADC1 MC2	ADC1 MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_ADC3R bits description.

27.5.67 HRTIM ADC trigger 2 register (HRTIM_ADC2R)

Address offset: 0x3C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2 TERST	ADC2 TEC4	ADC2 TEC3	ADC2 TEC2	ADC2 TDRST	ADC2 TDPER	ADC2 TDC4	ADC2 TFPER	ADC2T DC2	ADC2T CRST	ADC2T CPER	ADC2T CC4	ADC2 TFC4	ADC2T CC2	ADC2T BPER	ADC2T BC4
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**
ADC2 TFC3	ADC2 TBC2	ADC2 TAPER	ADC2 TAC4	ADC2 TFC2	ADC2 TAC2	ADC2 EEV10	ADC2 EEV9	ADC2 EEV8	ADC2 EEV7	ADC2 EEV6	ADC2 MPER	ADC2 MC4	ADC2 MC3	ADC2 MC2	ADC2 MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_ADC4R bits description.

27.5.68 HRTIM ADC trigger 3 register (HRTIM_ADC3R)

Address offset: 0x3C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC3 TEPER	ADC3 TEC4	ADC3 TEC3	ADC3 TFRST	ADC3 TDPER	ADC3 TDC4	ADC3 TDC3	ADC3 TFPER	ADC3 TCPER	ADC3 TCC4	ADC3 TCC3	ADC3 TFC4	ADC3 TBRST	ADC3 TBPER	ADC3 TBC4	ADC3 TBC3
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
ADC3 TFC3	ADC3 TARST	ADC3 TAPER	ADC3 TAC4	ADC3 TAC3	ADC3 TFC2	ADC3 EEV5	ADC3 EEV4	ADC3 EEV3	ADC3 EEV2	ADC3 EEV1	ADC3 MPER	ADC3 MC4	ADC3 MC3	ADC3 MC2	ADC3 MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC3TEPER**: ADC trigger 3 on timer E period
Refer to ADC3TAPER description.

Bit 30 **ADC3TEC4**: ADC trigger 3 on timer E compare 4
Refer to ADC3TFC2 description.

Bit 29 **ADC3TEC3**: ADC trigger 3 on timer E compare 3
Refer to ADC3TFC2 description.

Bit 28 **ADC3TFRST**: ADC trigger 3 on timer F reset and counter roll-over(1)
Refer to ADC3TARST description.

Bit 27 **ADC3TDPER**: ADC trigger 3 on timer D period
Refer to ADC3TAPER description.

Bit 26 **ADC3TDC4**: ADC trigger 3 on timer D compare 4
Refer to ADC3TFC2 description.

Bit 25 **ADC3TDC3**: ADC trigger 3 on timer D compare 3
Refer to ADC3TFC2 description.

Bit 24 **ADC3TFPER**: ADC trigger 3 on timer F period
Refer to ADC3TAPER description.

Bit 23 **ADC3TCPER**: ADC trigger 3 on timer C period
Refer to ADC3TAPER description.

Bit 22 **ADC3TCC4**: ADC trigger 3 on timer C compare 4
Refer to ADC3TFC2 description.

Bit 21 **ADC3TCC3**: ADC trigger 3 on timer C compare 3
Refer to ADC3TFC2 description.

Bit 20 **ADC3TFC4**: ADC trigger 3 on timer F compare 4
Refer to ADC3TAC2 description.

Bit 19 **ADC3TBRST**: ADC trigger 3 on timer B reset and counter roll-over(1)
Refer to ADC3TBRST description.

Bit 18 **ADC3TBPER**: ADC trigger 3 on timer B period
Refer to ADC3TAPER description.

Bit 17 **ADC3TBC4**: ADC trigger 3 on timer B compare 4
Refer to ADC3TFC2 description.

Bit 16 **ADC3TBC3**: ADC trigger 3 on timer B compare 3
Refer to ADC3TFC2 description.

Bit 15 **ADC3TFC3**: ADC trigger 3 on timer F compare 3
Refer to ADC3TFC2 description.

Bit 14 **ADC3TARST**: ADC trigger 3 on timer A reset and counter roll-over[(1)]
This bit enables the generation of an ADC trigger upon timer A reset and roll-over event, on ADC trigger 1 output.

Bit 13 **ADC3TAPER**: ADC trigger 3 on timer A period
This bit enables the generation of an ADC trigger upon timer A period event, on ADC trigger 1 output.

Bit 12 **ADC3TAC4**: ADC trigger 3 on timer A compare 4
Refer to ADC3TFC2 description.

Bit 11 **ADC3TAC3**: ADC trigger 3 on timer A compare 3
Refer to ADC3TFC2 description.

Bit 10 **ADC3TFC2**: ADC trigger 3 on timer F compare 2
This bit enables the generation of an ADC trigger upon timer F compare 2 event, on ADC trigger 3 output.

Bit 9 **ADC3EEV5**: ADC trigger 3 on external event 5[(1)]
Refer to ADC3EEV1 description.

Bit 8 **ADC3EEV4**: ADC trigger 3 on external event 4[(1)]
Refer to ADC3EEV1 description.

Bit 7 **ADC3EEV3**: ADC trigger 3 on external event 3[(1)]
Refer to ADC3EEV1 description.

Bit 6 **ADC3EEV2**: ADC trigger 3 on external event 2[(1)]
Refer to ADC3EEV1 description.

Bit 5 **ADC3EEV1**: ADC trigger 3 on external event 1[(1)]
This bit enables the generation of an ADC trigger upon external event 1, on ADC trigger 3 output.

Bit 4 **ADC3MPER**: ADC trigger 3 on master period
This bit enables the generation of an ADC trigger upon master timer period event, on ADC trigger 3 output.

Bit 3 **ADC3MC4**: ADC trigger 3 on master compare 4
Refer to ADC3MC1 description.

Bit 2 **ADC3MC3**: ADC trigger 3 on master compare 3
Refer to ADC3MC1 description.

Bit 1 **ADC3MC2**: ADC trigger 3 on master compare 2
Refer to ADC3MC1 description.

Bit 0 **ADC3MC1**: ADC trigger 3 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 3 output.

1. These triggers are differing from HRTIM_ADC2R/HRTIM_ADC4R to HRTIM_ADC1R/HRTIM_ADC3R.

27.5.69 HRTIM ADC trigger 4 register (HRTIM_ADC4R)

Address offset: 0x3C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC4 TERST	ADC4 TEC4	ADC4 TEC3	ADC4 TEC2	ADC4 TDRST	ADC4 TDPER	ADC4 TDC4	ADC4 TFPER	ADC4T DC2	ADC4T CRST	ADC4T CPER	ADC4T CC4	ADC4 TFC4	ADC4T CC2	ADC4T BPER	ADC4T BC4
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
ADC4 TFC3	ADC4 TBC2	ADC4 TAPER	ADC4 TAC4	ADC4 TFC2	ADC4 TAC2	ADC4 EEV10	ADC4 EEV9	ADC4 EEV8	ADC4 EEV7	ADC4 EEV6	ADC4 MPER	ADC4 MC4	ADC4 MC3	ADC4 MC2	ADC4 MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC4TERST**: ADC trigger 4 on timer E reset and counter roll-over(1)
Refer to ADC4TCRST description.

Bit 30 **ADC4TEC4**: ADC trigger 4 on timer E compare 4
Refer to ADC4TAC2 description.

Bit 29 **ADC4TEC3**: ADC trigger 4 on timer E compare 3
Refer to ADC4TAC2 description.

Bit 28 **ADC4TEC2**: ADC trigger 4 on timer E compare 2
Refer to ADC4TAC2 description.

Bit 27 **ADC4TDRST**: ADC trigger 4 on timer D reset and counter roll-over(1)
Refer to ADC4TCRST description.

Bit 26 **ADC4TDPER**: ADC trigger 4 on timer D period
Refer to ADC4TAPER description.

Bit 25 **ADC4TDC4**: ADC trigger 4 on timer D compare 4
Refer to ADC4TAC2 description.

Bit 24 **ADC4TFPER**: ADC trigger 4 on timer F period
Refer to ADC4TAPER description.

Bit 23 **ADC4TDC2**: ADC trigger 2 on timer D compare 2
Refer to ADC4TAC2 description.

Bit 22 **ADC4TCRST**: ADC trigger 4 on timer C reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer C reset and roll-over event, on ADC trigger 1 output.

Bit 21 **ADC4TCPER**: ADC trigger 4 on timer C period
Refer to ADC4TAPER description.

Bit 20 **ADC4TCC4**: ADC trigger 4 on timer C compare 4
Refer to ADC4TAC2 description.

Bit 19 **ADC4TFC4**: ADC trigger 4 on timer F compare 4
Refer to ADC4TAC2 description.

Bit 18 **ADC4TCC2**: ADC trigger 4 on timer C compare 2
Refer to ADC4TAC2 description.

Bit 17 **ADC4TBPER**: ADC trigger 4 on timer B period
Refer to ADC4TAPER description.

Bit 16 **ADC4TBC4**: ADC trigger 4 on timer B compare 4
Refer to ADC4TAC2 description.

Bit 15 **ADC4TFC3**: ADC trigger 4 on timer F compare 3
Refer to ADC4TAC2 description.

Bit 14 **ADC4TBC2**: ADC trigger 4 on timer B compare 2
Refer to ADC4TAC2 description.

Bit 13 **ADC4TAPER**: ADC trigger 4 on timer A period
This bit enables the generation of an ADC trigger upon timer A event, on ADC trigger 2 output.

Bit 12 **ADC4TAC4**: ADC trigger 4 on timer A compare 4
Refer to ADC4TAC2 description.

Bit 11 **ADC4TFC2**: ADC trigger 4 on timer F compare 2
Refer to ADC4TAC2 description.

Bit 10 **ADC4TAC2**: ADC trigger 4 on timer A compare 2
This bit enables the generation of an ADC trigger upon timer A compare 2, on ADC trigger 2 output.

Bit 9 **ADC4EEV10**: ADC trigger 4 on external event 10(1)
Refer to ADC4EEV6 description.

Bit 8 **ADC4EEV9**: ADC trigger 4 on external event 9(1)
Refer to ADC4EEV6 description.

Bit 7 **ADC4EEV8**: ADC trigger 4 on external event 8(1)
Refer to ADC4EEV6 description.

Bit 6 **ADC4EEV7**: ADC trigger 4 on external event 7(1)
Refer to ADC4EEV6 description.

Bit 5 **ADC4EEV6**: ADC trigger 4 on external event 6(1)
This bit enables the generation of an ADC trigger upon external event 6, on ADC trigger 2 output.

Bit 4 **ADC4MPER**: ADC trigger 4 on master period
This bit enables the generation of an ADC trigger upon master period event, on ADC trigger 2 output.

Bit 3 **ADC4MC4**: ADC trigger 4 on master compare 4
Refer to ADC4MC1 description.

Bit 2 **ADC4MC3**: ADC trigger 4 on master compare 3
Refer to ADC4MC1 description.

Bit 1 **ADC4MC2**: ADC trigger 4 on master compare 2
Refer to ADC4MC1 description.

Bit 0 **ADC4MC1**: ADC trigger 4 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 2 output.

1. These triggers are differing from HRTIM_ADC1R/HRTIM_ADC3R to HRTIM_ADC2R/HRTIM_ADC4R.

27.5.70 HRTIM DLL control register (HRTIM_DLLCR)

Address offset: 0x3CC

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.	CALRTE[1:0]		CALEN	CAL
												rw	rw	rw	wo

Bits 31:4 Reserved, must be kept at reset value

Bits 3:2 **CALRTE[1:0]**: DLL Calibration rate

This defines the DLL calibration periodicity.

00: 1048576 * t_{HRTIM} (6.168 ms for f_{HRTIM} = 170 MHz)
01: 131072 * t_{HRTIM} (771 µs for f_{HRTIM} = 170 MHz)
10: 16384 * t_{HRTIM} (96 µs for f_{HRTIM} = 170 MHz)
11: 2048 * t_{HRTIM} (12 µs for f_{HRTIM} = 170 MHz)

Bit 1 **CALEN**: DLL Calibration Enable

This bit enables the periodic DLL calibration, as per CALRTE[1:0] bit setting. When CALEN bit is reset, the calibration can be started in single-shot mode with CAL bit.

0: Periodic calibration disabled
1: Calibration is performed periodically, as per CALRTE[1:0] setting

Note: CALEN must not be set simultaneously with CAL bit

Bit 0 **CAL**: DLL Calibration Start

This bit starts the DLL calibration process. It is write-only.

0: No calibration request
1: Calibration start

Note: CAL must not be set simultaneously with CALEN bit

27.5.71 HRTIM fault input register 1 (HRTIM_FLTINR1)

Address offset: 0x3D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4LCK	FLT4F[3:0]				FLT4 SRC[0]	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3 SRC[0]	FLT3P	FLT3E
rwo	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**
FLT2LCK	FLT2F[3:0]				FLT2 SRC[0]	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1 SRC[0]	FLT1P	FLT1E
rwo	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4LCK**: Fault 4 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 30:27 **FLT4F[3:0]**: Fault 4 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 26 FLT4SRC[0]: Fault 4 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 25 **FLT4P**: Fault 4 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 24 **FLT4E**: Fault 4 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 23 **FLT3LCK**: Fault 3 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 22:19 **FLT3F[3:0]**: Fault 3 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 18 FLT3SRC[0]: Fault 3 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 17 **FLT3P**: Fault 3 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 16 **FLT3E**: Fault 3 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 15 **FLT2LCK**: Fault 2 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 14:11 **FLT2F[3:0]**: Fault 2 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 10 FLT2SRC[0]: Fault 2 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 9 **FLT2P**: Fault 2 polarity
Refer to FLT2P description in HRTIM_FLTINR2 register.

Bit 8 **FLT2E**: Fault 2 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 7 **FLT1LCK**: Fault 1 lock

Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 6:3 **FLT1F[3:0]**: Fault 1 filter

Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 2 **FLT1SRC[0]**: Fault 1 source bit 0

Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 1 **FLT1P**: Fault 1 polarity

Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 0 **FLT1E**: Fault 1 enable

Refer to FLT5E description in HRTIM_FLTINR2 register.

27.5.72 HRTIM fault input register 2 (HRTIM_FLTINR2)

Address offset: 0x3D4

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.	FLTSD[1:0]		Res.	Res.	FLT6 SRC[1]	FLT5 SRC[1]	FLT4 SRC[1]	FLT3 SRC[1]	FLT2 SRC[1]	FLT1 SRC[1]
						rw	rw			rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FLT6 LCK	FLT6F[3:0]				FLT6 SRC[0]	FLT6P	FLT6E	FLT5 LCK	FLT5F[3:0]				FLT5 SRC[0]	FLT5P	FLT5E
rwo	rw	rw	rw	rw	rw	rw	rw	rwo	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **FLTSD[1:0]**: Fault sampling clock division

This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the fault signal sampling clock (f_{FLTS}) used by the digital filters.

00: $f_{FLTS}=f_{HRTIM}$
01: $f_{FLTS}=f_{HRTIM} / 2$
10: $f_{FLTS}=f_{HRTIM} / 4$
11: $f_{FLTS}=f_{HRTIM} / 8$

Note: This bitfield must be written prior to any of the FLTxE enable bits.

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

Bit 21 **FLT6SRC[1]**: Fault 6 source bit 1

Refer to FLT5SRC[0] description.

Bit 20 **FLT5SRC[1]**: Fault 5 source bit 1

Refer to FLT5SRC[0] description.

Bit 19 **FLT4SRC[1]**: Fault 4 source bit 1

Refer to FLT5SRC[0] description.

Bit 18 **FLT3SRC[1]**: Fault 3 source bit 1

Refer to FLT5SRC[0] description.

Bit 17 **FLT2SRC[1]**: Fault 2 source bit 1

Refer to FLT5SRC[0] description.

Bit 16 **FLT1SRC[1]**: Fault 1 source bit 1

Refer to FLT5SRC[0] description.

Bit 15 **FLT6LCK**: Fault 6 lock

Refer to FLT5LCK description.

Bits 14:11 **FLT6F[3:0]**: Fault 6 filter

Refer to FLT5F[3:0] description.

Bit 10 **FLT6SRC[0]**: Fault 6 source bit 0

Refer to FLT5SRC[0] description.

Bit 9 **FLT6P**: Fault 6 polarity

Refer to FLT5P description.

Bit 8 **FLT6E**: Fault 6 enable

Refer to FLT5E description.

Bit 7 **FLT5LCK**: Fault 5 lock

The FLT5LCK bit modifies the write attributes of the fault programming bit, so that they are protected against spurious write accesses.

This bit is write-once. Once it has been set, it cannot be modified till the next system reset.

0: FLT5E, FLT5P, FLT5SRC[1:0], FLT5F[3:0] and FLT5BLK bits are read/write.

1: FLT5E, FLT5P, FLT5SRC[1:0], FLT5F[3:0] and FLT5BLK bits cannot be written (read-only mode)

Bits 6:3 **FLT5F[3:0]**: Fault 5 filter

This bitfield defines the frequency used to sample FLT5 input and the length of the digital filter applied to FLT5. The digital filter is made of an event counter in which N events are needed to validate a transition on the output.

0000: No filter, FLT5 acts asynchronously

0001: $f_{SAMPLING} = f_{HRTIM}$, N = 2

0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4

0011: $f_{SAMPLING} = f_{HRTIM}$, N = 8

0100: $f_{SAMPLING} = f_{FLTS}/2$, N = 6

0101: $f_{SAMPLING} = f_{FLTS}/2$, N = 8

0110: $f_{SAMPLING} = f_{FLTS}/4$, N = 6

0111: $f_{SAMPLING} = f_{FLTS}/4$, N = 8

1000: $f_{SAMPLING} = f_{FLTS}/8$, N = 6

1001: $f_{SAMPLING} = f_{FLTS}/8$, N = 8

1010: $f_{SAMPLING} = f_{FLTS}/16$, N = 5

1011: $f_{SAMPLING} = f_{FLTS}/16$, N = 6

1100: $f_{SAMPLING} = f_{FLTS}/16$, N = 8

1101: $f_{SAMPLING} = f_{FLTS}/32$, N = 5

1110: $f_{SAMPLING} = f_{FLTS}/32$, N = 6

1111: $f_{SAMPLING} = f_{FLTS}/32$, N = 8

Note: This bitfield is written only when FLT5E enable bit is reset.

This bitfield is modified when FLT5LOCK has been programmed.

Bit 2 **FLT5SRC[0]**: Fault 5 source bit 0

The FTL5SRC[1:0] bitfield selects the FAULT5 input source (refer to *Table 233* for connection details).

00: Fault 5 input is HRTIM_FLT5 input pin

01: Fault 5 input is connected to a COMPx output

10: Fault 5 input is EEV5_muxout input pin

01: Reserved

Note: This bitfield is written only when FLT5E enable bit is reset.

Bit 1 **FLT5P**: Fault 5 polarity

This bit selects the FAULT5 input polarity.

0: Fault 5 input is active low

1: Fault 5 input is active high

Note: This bitfield is written only when FLT5E enable bit is reset.

Bit 0 **FLT5E**: Fault 5 enable

This bit enables the global FAULT5 input circuitry.

0: Fault 5 input disabled

1: Fault 5 input enabled

27.5.73 HRTIM burst DMA master timer update register (HRTIM_BDMUPR)

Address offset: 0x3D8

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.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MCMP4**: MCMP4R register update enable

Refer to MCR description.

Bit 8 **MCMP3**: MCMP3R register update enable

Refer to MCR description.

Bit 7 **MCMP2**: MCMP2R register update enable

Refer to MCR description.

Bit 6 **MCMP1**: MCMP1R register update enable

Refer to MCR description.

Bit 5 **MREP**: MREP register update enable

Refer to MCR description.

Bit 4 **MPER**: MPER register update enable

Refer to MCR description.

Bit 3 **MCNT**: MCNTR register update enable

Refer to MCR description.

Bit 2 **MDIER**: MDIER register update enable

Refer to MCR description.

Bit 1 **MICR**: MICR register update enable

Refer to MCR description.

Bit 0 **MCR**: MCR register update enable

This bit defines if the master timer MCR register is part of the list of registers to be updated by the burst DMA.

0: MCR register is not updated by burst DMA accesses

1: MCR register is updated by burst DMA accesses

27.5.74 HRTIM burst DMA timer x update register (HRTIM_BDTxUPR) (x = A to F)

Address offset: Block A: 0x3DC

Address offset: Block B: 0x3E0

Address offset: Block C: 0x3E4

Address offset: Block D: 0x3E8

Address offset: Block E: 0x3EC

Address offset: Block F: 0x3F4

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.	TIMxEEFR3	TIMxCR2	TIMxFLTR	TIMxOUTR	TIMxCHPR	TIMxRSTR	TIMxEEFR2
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMxEEFR1	TIMxRST2R	TIMxSET2R	TIMxRST1R	TIMxSET1R	TIMxDTxR	TIMxCMP4	TIMxCMP3	TIMxCMP2	TIMxCMP1	TIMxREP	TIMxPER	TIMxCNT	TIMxDIER	TIMxICR	TIMxCR
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 **TIMxEEFR3**: HRTIM_EEFxR3 register update enable
Refer to TIMxCR description.

Bit 21 **TIMxCR2**: HRTIM_TIMxCR2 register update enable
Refer to TIMxCR description.

Bit 20 **TIMxFLTR**: HRTIM_FLTxR register update enable
Refer to TIMxCR description.

Bit 19 **TIMxOUTR**: HRTIM_OUTxR register update enable
Refer to TIMxCR description.

Bit 18 **TIMxCHPR**: HRTIM_CHPxR register update enable
Refer to TIMxCR description.

Bit 17 **TIMxRSTR**: HRTIM_RSTxR register update enable
Refer to TIMxCR description.

Bit 16 **TIMxEEFR2**: HRTIM_EEFxR2 register update enable
Refer to TIMxCR description.

Bit 15 **TIMxEEFR1**: HRTIM_EEFxR1 register update enable
Refer to TIMxCR description.

Bit 14 **TIMxRST2R**: HRTIM_RST2xR register update enable
Refer to TIMxCR description.

Bit 13 **TIMxSET2R**: HRTIM_SET2xR register update enable
Refer to TIMxCR description.

Bit 12 **TIMxRST1R**: HRTIM_RST1xR register update enable
Refer to TIMxCR description.

Bit 11 **TIMxSET1R**: HRTIM_SET1xR register update enable
Refer to TIMxCR description.

Bit 10 **TIMxDTR**: HRTIM_DTxR register update enable
Refer to TIMxCR description.

Bit 9 **TIMxCMP4**: HRTIM_CMP4xR register update enable
Refer to TIMxCR description.

Bit 8 **TIMxCMP3**: HRTIM_CMP3xR register update enable
Refer to TIMxCR description.

Bit 7 **TIMxCMP2**: HRTIM_CMP2xR register update enable
Refer to TIMxCR description.

Bit 6 **TIMxCMP1**: HRTIM_CMP1xR register update enable
Refer to TIMxCR description.

Bit 5 **TIMxREP**: HRTIM_REPxR register update enable
Refer to TIMxCR description.

Bit 4 **TIMxPER**: HRTIM_PERxR register update enable
Refer to TIMxCR description.

Bit 3 **TIMxCNT**: HRTIM_CNTxR register update enable
Refer to TIMxCR description.

Bit 2 **TIMxDIER**: HRTIM_TIMxDIER register update enable
Refer to TIMxCR description.

Bit 1 **TIMxICR**: HRTIM_TIMxICR register update enable
Refer to TIMxCR description.

Bit 0 **TIMxCR**: HRTIM_TIMxCR register update enable
This bit defines if the master timer MCR register is part of the list of registers to be updated by the burst DMA.
0: HRTIM_TIMxCR register is not updated by burst DMA accesses
1: HRTIM_TIMxCR register is updated by burst DMA accesses

27.5.75 HRTIM burst DMA data register (HRTIM_BDMADR)

Address offset: 0x3F0

Reset value: 0x0000 0000

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

Bits 31:0 **BDMADR[31:0]**: Burst DMA data register
Write accesses to this register triggers:
- the copy of the data value into the registers enabled in BDTxUPR and BDMUPR register bits
- the increment of the register pointer to the next location to be filled

27.5.76 HRTIM ADC extended trigger register (HRTIM_ADCER)

Address offset: 0x3F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ADC10TRG[4:0]					ADC9TRG[4:0]					ADC8TRG[4:0]				
	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.	ADC7TRG[4:0]					ADC6TRG[4:0]					ADC5TRG[4:0]				
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **ADC10TRG[4:0]**: ADC trigger 10 selection

This bit selects the ADC trigger 10 source.
Refer to ADC6TRG[4:0] description.

Bits 25:21 **ADC9TRG[4:0]**: ADC trigger 9 selection

This bit selects the ADC trigger 9 source.
Refer to ADC5TRG[4:0] description.

Bits 20:16 **ADC8TRG[4:0]**: ADC trigger 8 selection

This bit selects the ADC trigger 8 source.
Refer to ADC6TRG[4:0] description.

Bit 15 Reserved, must be kept at reset value.

Bits 14:10 **ADC7TRG[4:0]**: ADC trigger 7 selection

This bit selects the ADC trigger 7 source.
Refer to ADC5TRG[4:0] description.

Bits 9:5 **ADC6TRG[4:0]**: ADC trigger 6 selection

This bit selects the ADC trigger 6 source.
0: Trigger on master compare 1
1: Trigger on master compare 2
2: Trigger on master compare 3
3: Trigger on master compare 4
4: Trigger on master period
5: Trigger on external event 6
6: Trigger on external event 7
7: Trigger on external event 8
8: Trigger on external event 9
9: Trigger on external event 10
10: Trigger on timer A compare 2
11: Trigger on timer A compare 4
12: Trigger on timer A period
13: Trigger on timer B compare 2
14: Trigger on timer B compare 4
15: Trigger on timer B period
16: Trigger on timer C compare 2
17: Trigger on timer C compare 4
18: Trigger on timer C period
19: Trigger on timer C reset and counter roll-over
20: Trigger on timer D compare 2
21: Trigger on timer D compare 4
22: Trigger on timer D period
23: Trigger on timer D reset and counter roll-over
24: Trigger on timer E compare 2
25: Trigger on timer E compare 3
26: Trigger on timer E compare 4
27: Trigger on timer E reset and counter roll-over
28: Trigger on timer F compare 2
29: Trigger on timer F compare 3
30: Trigger on timer F compare 4
31: Trigger on timer F period

Bits 4:0 **ADC5TRG[4:0]**: ADC trigger 5 selection

This bit selects the ADC trigger 5 source.

0: Trigger on master compare 1
1: Trigger on master compare 2
2: Trigger on master compare 3
3: Trigger on master compare 4
4: Trigger on master period
5: Trigger on external event 1
6: Trigger on external event 2
7: Trigger on external event 3
8: Trigger on external event 4
9: Trigger on external event 5
10: Trigger on timer A compare 3
11: Trigger on timer A compare 4
12: Trigger on timer A period
13: Trigger on timer A reset and counter roll-over
14: Trigger on timer B compare 3
15: Trigger on timer B compare 4
16: Trigger on timer B period
17: Trigger on timer B reset and counter roll-over
18: Trigger on timer C compare 3
19: Trigger on timer C compare 4
20: Trigger on timer C period
21: Trigger on timer D compare 3
22: Trigger on timer D compare 4
23: Trigger on timer D period
24: Trigger on timer E compare 3
25: Trigger on timer E compare 4
26: Trigger on timer E period
27: Trigger on timer F compare 2
28: Trigger on timer F compare 3
29: Trigger on timer F compare 4
30: Trigger on timer F period
31: Trigger on timer F reset and counter roll-over

27.5.77 HRTIM ADC trigger update register (HRTIM_ADCUR)

Address offset: 0x3FC

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.	AD10USRC[2:0]			Res.	AD9USRC[2:0]		
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.	AD8USRC[2:0]			Res.	AD7USRC[2:0]			Res.	AD6USRC[2:0]			Res.	AD5USRC[2:0]		
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.

Bits 22:20 **AD10USRC[2:0]**: ADC trigger 10 update source
Refer to AD5USRC[2:0] description.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **AD9USRC[2:0]**: ADC trigger 9 update source
Refer to AD5USRC[2:0] description.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **AD8USRC[2:0]**: ADC trigger 8 update source
Refer to AD5USRC[2:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **AD7USRC[2:0]**: ADC trigger 7 update source
Refer to AD5USRC[2:0] description.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **AD6USRC[2:0]**: ADC trigger 6 update source
Refer to AD5USRC[2:0] description.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **AD5USRC[2:0]**: ADC trigger 5 update source

These bits define the source which triggers the update of the ADC5TRG[4:0] bitfield in the HRTIM_ADCER register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, with the BRSTDMA[1:0] bitfield in HRTIM_MCR or the UPDGAT[3:0] bitfield in HRTIM_TIMxCR register.

000: Master timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110: Timer F
111: Reserved

27.5.78 HRTIM ADC post scaler register 1 (HRTIM_ADCPS1)

Address offset: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC5PSC[4:0]					Res.	ADC4PSC[4:0]					Res.	ADC3PSC[4]
			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
ADC3PSC[3:0]				Res.	ADC2PSC[4:0]					Res.	ADC1PSC[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADC5PSC[4:0]**: ADC 5 post scaler
Refer to ADC1PSC[4:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **ADC4PSC[4:0]**: ADC 4 post scaler
Refer to ADC1PSC[4:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **ADC3PSC[4:0]**: ADC 3 post scaler
Refer to ADC1PSC[4:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **ADC2PSC[4:0]**: ADC 2 post scaler
Refer to ADC1PSC[4:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **ADC1PSC[4:0]**: ADC 1 post scaler
This bit selects the ADC 1 Post scaler ratio.

27.5.79 HRTIM ADC post scaler register 2 (HRTIM_ADCPS2)

Address offset: 0x404

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC10PSC[4:0]					Res.	ADC9PSC[4:0]					Res.	ADC8PSC[4]
			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
ADC8PSC[3:0]				Res.	ADC7PSC[4:0]					Res.	ADC6PSC[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADC10PSC[4:0]**: ADC 10 post scaler
Refer to ADC1PSC[4:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **ADC9PSC[4:0]**: ADC 9 post scaler
Refer to ADC1PSC[4:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **ADC8PSC[4:0]**: ADC 8 post scaler
Refer to ADC1PSC[4:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **ADC7PSC[4:0]**: ADC 7 post scaler
Refer to ADC1PSC[4:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **ADC6PSC[4:0]**: ADC 6 post scaler
Refer to ADC1PSC[4:0] description.

27.5.80 HRTIM fault input register 3 (HRTIM_FLTINR3)

Address offset: 0x408

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4 RSTM	FLT4 CRES	FLT4CNT[3:0]				FLT4 BLKS	FLT4 BLKE	FLT3 RSTM	FLT3 CRES	FLT3CNT[3:0]				FLT3 BLKS	FLT3 BLKE
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
FLT2 RSTM	FLT2 CRES	FLT2CNT[3:0]				FLT2 BLKS	FLT2 BLKE	FLT1 RSTM	FLT1 CRES	FLT1CNT[3:0]				FLT1 BLKS	FLT1 BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4RSTM**: Fault 4 reset mode

Refer to FLT5RSTM description.

Bit 30 **FLT4CRES**: Fault 4 counter reset

Refer to FLT5CRES description.

Bits 29:26 **FLT4CNT[3:0]**: Fault 4 counter

Refer to FLT5CNT description.

Bit 25 **FLT4BLKS**: Fault 4 blanking source

Refer to FLT5BLKS description.

Bit 24 **FLT4BLKE**: Fault 4 blanking enable

Refer to FLT5BLKE description.

Bit 23 **FLT3RSTM**: Fault 3 reset mode

Refer to FLT5RSTM description.

Bit 22 **FLT3CRES**: Fault 3 counter reset

Refer to FLT5CRES description.

Bits 21:18 **FLT3CNT[3:0]**: Fault 3 counter

Refer to FLT5CNT description.

Bit 17 **FLT3BLKS**: Fault 3 blanking source

Refer to FLT5BLKS description.

Bit 16 **FLT3BLKE**: Fault 3 blanking enable

Refer to FLT5BLKE description.

Bit 15 **FLT2RSTM**: Fault 2 reset mode

Refer to FLT5RSTM description.

Bit 14 **FLT2CRES**: Fault 2 counter reset

Refer to FLT5CRES description.

Bits 13:10 **FLT2CNT[3:0]**: Fault 2 counter

Refer to FLT5CNT description.

Bit 9 **FLT2BLKS**: Fault 2 blanking source

Refer to FLT5BLKS description.

Bit 8 **FLT2BLKE**: Fault 2 blanking enable

Refer to FLT5BLKE description.

Bit 7 **FLT1RSTM**: Fault 1 reset mode

Refer to FLT5RSTM description.

Bit 6 **FLT1CRES**: Fault 1 counter reset

Refer to FLT5CRES description.

Bits 5:2 **FLT1CNT[3:0]**: Fault 1 counter

Refer to FLT5CNT description.

Bit 1 **FLT1BLKS**: Fault 1 blanking source

Refer to FLT5BLKS description.

Bit 0 **FLT1BLKE**: Fault 1 blanking enable

Refer to FLT5BLKE description.

27.5.81 HRTIM fault input register 4 (HRTIM_FLTINR4)

Address offset: 0x40C

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**
FLT6 RSTM	FLT6 CRES	FLT6CNT[3:0]				FLT6 BLKS	FLT6 BLKE	FLT5 RSTM	FLT5 CRES	FLT5CNT[3:0]				FLT5 BLKS	FLT5 BLKE
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.

Bit 15 **FLT6RSTM**: Fault 6 reset mode

Refer to FLT5RSTM description.

Bit 14 **FLT6CRES**: Fault 6 counter reset

Refer to FLT5CRES description.

Bits 13:10 **FLT6CNT[3:0]**: Fault 6 counter

Refer to FLT5CNT description.

Bit 9 **FLT6BLKS**: Fault 6 blanking source

Refer to FLT5BLKS description.

Bit 8 **FLT6BLKE**: Fault 6 blanking enable

Refer to FLT6BLKE description.

Bit 7 **FLT5RSTM**: Fault 5 reset mode

This bit selects the FAULT5 counter reset mode

0: Fault 5 counter is reset on each reset / roll-over event

1: Fault 5 counter is reset on each reset / roll-over event only if no fault occurred during last counting period.

This bitfield is written only when FLT5E enable bit is reset.

Bit 6 **FLT5CRES**: Fault 5 counter reset

This bit resets the FAULT5 counter. It is set by software and reset by hardware.

0: No action

1: Fault 5 counter is reset

Bits 5:2 **FLT5CNT[3:0]**: Fault 5 counter

This bitfield selects the FAULT5 counter threshold. A fault is considered valid when the number of events is equal to the (FLT5CNT[3:0]+1) value.

Bit 1 **FLT5BLKS**: Fault 5 blanking source

The FTL5BLKS bit selects the FAULT5 blanking source (refer to *Table 235* for details).

0: Fault 5 reset-aligned blanking window

1: Fault 5 Moving blanking window

Note: This bitfield is written only when FLT5E enable bit is reset.

Bit 0 **FLT5BLKE**: Fault 5 blanking enable

The FTL5BLKE bit selects the FAULT5 blanking mode. The blanking source is defined by the FLT5BLKS bit.

0: No blanking on fault 5

1: Fault 5 blanking mode

Note: This bitfield is written only when FLT5E enable bit is reset

27.5.82 HRTIM register map

The tables below summarize the HRTIM registers mapping.

Table 241. HRTIM Register map and reset values – master timer

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	HRTIM_MCR	BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	TFCEN	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN	SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		INTLVD[1:0]		HALF	RETRIG	CONT	CKPSC[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
0x004	HRTIM_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.	MUPD	SYNC	MREP	MCMP4	MCMP3	MCMP2	MCMP1
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
0x008	HRTIM_MICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
0x00C	HRTIM_ MDIER(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
	Reset value	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
0x010	HRTIM_MCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCNT[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	HRTIM_MPER(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPER[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x018	HRTIM_MREP(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MREP[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x01C	HRTIM_ MCMP1R(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP1[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	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.
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x024	HRTIM_ MCMP2R(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP2[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	HRTIM_ MCMP3R(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP3[15:0]															
	Reserved value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	HRTIM_ MCMP4R(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

[1] This register can be preloaded (see *Table 226 on page 908*).

Table 242. HRTIM register map and reset values – TIMx (x= A..F)

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
0x080 (x=A) 0x100 (x=B) 0x180 (x=C) 0x200 (x=D) 0x280 (x=E) 0x300 (x=F)	HRTIM_ TIMxCR	UPDGAT [3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	TAU	TxRSTU	TxREPU	TFU	DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	RSYNCU	INTLVD[1:0]		PSHPLL	HALF	RETRIG	CONT	CKPSCx[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	0	0
0x084 (x=A) 0x104 (x=B) 0x184 (x=C) 0x204 (x=D) 0x284 (x=E) 0x304 (x=F)	HRTIM_ TIMxISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	O2CPY	O1CPY	O2STAT	O1STAT	IPPSTAT	CPPSTAT	Res.	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	Res.	REP	CMP4	CMP3	CMP2	CMP1
	Reset value	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0
0x088 (x=A) 0x108 (x=B) 0x188 (x=C) 0x208 (x=D) 0x288 (x=E) 0x308 (x=F)	HRTIM_ TIMxICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLYPRTC	RSTC	RSTx2C	SET2xC	RSTx1C	SET1xC	CPT2C	CPT1C	UPDC	Res.	REPC	CMP4C	CMP3C	CMP2C	CMP1C
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0
0x08C (x=A) 0x10C (x=B) 0x18C (x=C) 0x20C (x=D) 0x28C (x=E) 0x30C (x=F)	HRTIM_ TIMxDIER[1]	Res.	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SET1xDE	CPT2DE	CPT1DE	UPDDE	Res.	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE	Res.	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SET1xIE	CPT2IE	CPT1IE	UPDIE	Res.	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
	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
0x090 (x=A) 0x110 (x=B) 0x190 (x=C) 0x210 (x=D) 0x290 (x=E) 0x310 (x=F)	HRTIM_ CNTxR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTx[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094 (x=A) 0x114 (x=B) 0x194 (x=C) 0x214 (x=D) 0x294 (x=E) 0x314 (x=F)	HRTIM_ PERxR[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERx[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x098 (x=A) 0x118 (x=B) 0x198 (x=C) 0x218 (x=D) 0x298 (x=E) 0x318 (x=F)	HRTIM_ REPxR[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x09C (x=A) 0x11C (x=B) 0x19C (x=C) 0x21C (x=D) 0x29C (x=E) 0x31C (x=F)	HRTIM_ CMP1xR[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP1x[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0 (x=A) 0x120 (x=B) 0x1A0 (x=C) 0x220 (x=D) 0x2A0 (x=E) 0x320 (x=F)	HRTIM_ CMP1CxR[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]								CMP1x[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

Table 242. HRTIM register map and reset values – TIMx (x= A..F) (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
0x0A4 (x=A) 0x124 (x=B) 0x1A4 (x=C) 0x224 (x=D) 0x2A4 (x=E) 0x324 (x=F)	HRTIM_CMP2xR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP2x[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8 (x=A) 0x128 (x=B) 0x1A8 (x=C) 0x228 (x=D) 0x2A8 (x=E) 0x328 (x=F)	HRTIM_CMP3xR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP3x[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC (x=A) 0x12C (x=B) 0x1AC (x=C) 0x22C (x=D) 0x2AC (x=E) 0x32C (x=F)	HRTIM_CMP4xR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP4x[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0 (x=A) 0x130 (x=B) 0x1B0 (x=C) 0x230 (x=D) 0x2B0 (x=E) 0x330 (x=F)	HRTIM_CPT1xR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR	CPT1x[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4 (x=A) 0x134 (x=B) 0x1B4 (x=C) 0x234 (x=D) 0x2B4 (x=E) 0x334 (x=F)	HRTIM_CPT2xR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR	CPT2x[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8 (x=A) 0x138 (x=B) 0x1B8 (x=C) 0x238 (x=D) 0x2B8 (x=E) 0x338 (x=F)	HRTIM_DTxR(1)	DTFLKx	DTFSLKx	Res.	Res.	Res.	Res.	SDTFx	DTFx[8:0]									DTRLKx	DTRSLKx	Res.	DTPRSC[2:0]			SDTRx	DTRx[8:0]								
	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
0x0BC (x=A) 0x13C (x=B) 0x1BC (x=C) 0x23C (x=D) 0x2BC (x=E) 0x33C (x=F)	HRTIM_SETx1R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
	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
0x0C0 (x=A) 0x140 (x=B) 0x1C0 (x=C) 0x240 (x=D) 0x2C0 (x=E) 0x340 (x=F)	HRTIM_RSTx1R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
	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
0x0C4 (x=A) 0x144 (x=B) 0x1C4 (x=C) 0x244 (x=D) 0x2C4 (x=E) 0x344 (x=F)	HRTIM_SETx2R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
	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
0x0C8 (x=A) 0x148 (x=B) 0x1C8 (x=C) 0x248 (x=D) 0x2C8 (x=E) 0x348 (x=F)	HRTIM_RSTx2R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
	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 242. HRTIM register map and reset values – TIMx (x= A..F) (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
0x0CC (x=A) 0x14C (x=B) 0x1CC (x=C) 0x24C (x=D) 0x2CC (x=E) 0x34C (x=F)	**HRTIM_ EEFxR1**	Res.	Res.	Res.	EE5FLTR[3:0]				EE5LTCH	Res.	EE4FLTR[3:0]				EE4LTCH	Res.	EE3FLTR[3:0]				EE3LTCH	Res.	EE2FLTR[3:0]				EE2LTCH	Res.	EE1FLTR[3:0]				EE1LTCH
	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
0x0D0 (x=A) 0x150 (x=B) 0x1D0 (x=C) 0x250 (x=D) 0x2D0 (x=E) 0x350 (x=F)	**HRTIM_ EEFxR2**	Res.	Res.	Res.	EE10FLTR[3:0]				EE10LTCH	Res.	EE9FLTR[3:0]				EE9LTCH	Res.	EE8FLTR[3:0]				EE8LTCH	Res.	EE7FLTR[3:0]				EE7LTCH	Res.	EE6FLTR[3:0]				EE6LTCH
	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
0x0D4	**HRTIM_ RSTAR[1]**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	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
0x154	**HRTIM_ RSTBR[1]**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	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
0x1D4	**HRTIM_ RSTCR[1]**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	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
0x254	**HRTIM_ RSTDR[1]**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	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
0x2D4	**HRTIM_ RSTER[1]**	TIMFCMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	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
0x354	**HRTIM_ RSTFR[1]**	TIMECMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMECMP1
	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
0x0D8 (x=A) 0x158 (x=B) 0x1D8 (x=C) 0x258 (x=D) 0x2D8 (x=E) 0x358 (x=F)	**HRTIM_ CHPxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRTPW[3:0]				CARDTY[2:0]			CARFRQ[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Table 242. HRTIM register map and reset values – TIMx (x= A..F) (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
0x0DC	**HRTIM_ CPT1ACR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x15C	**HRTIM_ CPT1BCR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x1DC	**HRTIM_ CPT1CCR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x25C	**HRTIM_ CPT1DCR**	TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x2DC	**HRTIM_ CPT1ECR**	TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x35C	**HRTIM_ CPT1FCR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x0E0	**HRTIM_ CPT2ACR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x160	**HRTIM_ CPT2BCR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x1E0	**HRTIM_ CPT2CCR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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 242. HRTIM register map and reset values – TIMx (x= A..F) (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
0x260	**HRTIM_ CPT2DCR**	TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x2E0	**HRTIM_ CPT2ECR**	TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x360	**HRTIM_ CPT2FCR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	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
0x0E4 (x=A) 0x164 (x=B) 0x1E4 (x=C) 0x264 (x=D) 0x2E4 (x=E) 0x364 (x=F)	**HRTIM_ OUTxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM2	POL2	Res.	Res.	Res.	Res.	DLYPRT[2:0]			DLYPRTEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM1	POL1	Res.
	Reset value	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	-
0x0E8 (x=A) 0x168 (x=B) 0x1E8 (x=C) 0x268 (x=D) 0x2E8 (x=E) 0x368 (x=F)	**HRTIM_ FLTxR**	FLTLCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT6EN	FLT5EN	FLT4EN	FLT3EN	FLT2EN	FLT1EN
	Reset value	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0
0x0EC (x=A) 0x16C (x=B) 0x1EC (x=C) 0x26C (x=D) 0x2EC (x=E) 0x36C (x=F)	**HRTIM_ TIMxCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGHLF	Res.	Res.	GTCMP3	GTCMP1	FE ROM [1:0]		BM ROM [1:0]		AD ROM [1:0]		OUT ROM [1:0]		ROM [1:0]		Res.	UDM	Res.	DCDR	DCDS	DCDE
	Reset value	-	-	-	-	-	-	-	-	-	-	-	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0
0x0F0 (x=A) 0x170 (x=B) 0x1F0 (x=C) 0x270 (x=D) 0x2F0 (x=E) 0x370 (x=F)	**HRTIM_ TIMxEEFR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EEVACNT[5:0]						EEVASEL [5:0]				Res.	EEVARSTM	EEVACRES	EEVACE
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	-	0	0	0

1. This register can be preloaded (see *Table 226 on page 908*).

Table 243. HRTIM register map and reset values – common functions

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
0x0380	HRTIM_CR1	Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFUDIS	TEUDIS	TDUDIS	TCUDIS	TBUDIS	TAUDIS	MUDIS
	Reset value	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
0x0384	HRTIM_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWPF	SWPE	SWPD	SWPC	SWPB	SWPA	Res.	TFRST	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	TFSWU	TESWU	TDSWU	TCSWU	TBSWU	TASWU	MSWU
	Reset value	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	-	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0
0x388	HRTIM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	DLLRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT6	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
0x38C	HRTIM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	DLLRDYC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT6C	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
0x390	HRTIM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	DLLRDYIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT6IE	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
0x394	HRTIM_OENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF2OEN	TF1OEN	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
0x398	HRTIM_ODISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF2ODIS	TF1ODIS	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
0x39C	HRTIM_ODSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF2ODS	TF1ODS	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
0x3A0	HRTIM_BMCR	BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFBM	TEBM	TDBM	TCBM	TBBM	TABM	MTBM	Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
	Reset value	0	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
0x3A4	HRTIM_BMTRG	OCHPEV	EEV8	EEV7	TDEEV8	TAEEV7	TECMP2	TECMP1	TEREP	TFCMP1	TDCMP2	TFREP	TDREP	TDRST	TFRST	TCCMP1	TCREP	TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW
	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 243. HRTIM register map and reset values – common functions (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
0x3A8	HRTIM_BMCMPR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMCMP[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3AC	HRTIM_BMPER(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B0	HRTIM_EECR1	Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1:0]		EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[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	0	0	0	0	0	0
0x3B4	HRTIM_EECR2	Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1:0]		EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[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	0
0x3B8	HRTIM_EECR3	EEVSD[1:0]		Res.	Res.	EE10F[3:0]				Res.	Res.	EE9F[3:0]				Res.	Res.	EE8F[3:0]				Res.	Res.	EE7F[3:0]				Res.	Res.	EE6F[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
0x3BC	HRTIM_ADC1R(1)	ADC1TEPER	ADC1TEC4	ADC1TEC3	ADC1TFRST	ADC1TDPER	ADC1TDC4	ADC1TDC3	ADC1TFPER	ADC1TCPER	ADC1TCC4	ADC1TCC3	ADC1TFC4	ADC1TBRST	ADC1TBPER	ADC1TBC4	ADC1TBC3	ADC1TFC3	ADC1TARST	ADC1TAPER	ADC1TAC4	ADC1TAC3	ADC1TFC2	ADC1EEV5	ADC1EEV4	ADC1EEV3	ADC1EEV2	ADC1EEV1	ADC1MPER	ADC1MC4	ADC1MC3	ADC1MC2	ADC1MC1
	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
0x3C0	HRTIM_ADC2R(1)	ADC2TERST	ADC2TEC4	ADC2TEC3	ADC2TEC2	ADC2TDRST	ADC2TDPER	ADC2TDC4	ADC2TFPER	ADC2TDC2	ADC2TCRST	ADC2TCPER	ADC2TCC4	ADC2TFC4	ADC2TCC2	ADC2TBPER	ADC2TBC4	ADC2TFC3	ADC2TBC2	ADC2TAPER	ADC2TAC4	ADC2TFC2	ADC2TAC2	ADC2EEV10	ADC2EEV9	ADC2EEV8	ADC2EEV7	ADC2EEV6	ADC2MPER	ADC2MC4	ADC2MC3	ADC2MC2	ADC2MC1
	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
0x3C4	HRTIM_ADC3R(1)	ADC3TEPER	ADC3TEC4	ADC3TEC3	ADC3TFRST	ADC3TDPER	ADC3TDC4	ADC3TDC3	ADC3TFPER	ADC3TCPER	ADC3TCC4	ADC3TCC3	ADC3TFC4	ADC3TBRST	ADC3TBPER	ADC3TBC4	ADC3TBC3	ADC3TFC3	ADC3TARST	ADC3TAPER	ADC3TAC4	ADC3TAC3	ADC3TFC2	ADC3EEV5	ADC3EEV4	ADC3EEV3	ADC3EEV2	ADC3EEV1	ADC3MPER	ADC3MC4	ADC3MC3	ADC3MC2	ADC3MC1
	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
0x3C8	HRTIM_ADC4R(1)	ADC4TERST	ADC4TEC4	ADC4TEC3	ADC4TEC2	ADC4TDRST	ADC4TDPER	ADC4TDC4	ADC4TFPER	ADC4TDC2	ADC4TCRST	ADC4TCPER	ADC4TCC4	ADC4TFC4	ADC4TCC2	ADC4TBPER	ADC4TBC4	ADC4TFC3	ADC4TBC2	ADC4TAPER	ADC4TAC4	ADC4TFC2	ADC4TAC2	ADC4EEV10	ADC4EEV9	ADC4EEV8	ADC4EEV7	ADC4EEV6	ADC4MPER	ADC4MC4	ADC4MC3	ADC4MC2	ADC4MC1
	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
0x03CC	HRTIM_DLLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALRTE [1:0]		CALEN	CAL
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0

Table 243. HRTIM register map and reset values – common functions (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
0x3D0	HRTIM_ FLTINxR1	FLT4LCK	FLT4F[3:0]				FLT4SRC[0]	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3SRC[0]	FLT3P	FLT3E	FLT2LCK	FLT2F[3:0]				FLT2SRC[0]	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC[0]	FLT1P	FLT1E
	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
0x3D4	HRTIM_ FLTINxR2	Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		Res.	Res.	FLT6SRC[1]	FLT5SRC[1]	FLT4SRC[1]	FLT3SRC[1]	FLT2SRC[1]	FLT1SRC[1]	FLT6LCK	FLT6F[3:0]				FLT6SRC[0]	FLT6P	FLT6E	FLT5LCK	FLT5F[3:0]				FLT5SRC[0]	FLT5P	FLT5E
	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
0x3D8	HRTIM_ BDMUPDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0
0x3DC	HRTIM_ BDTAUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMAEEFR3	TIMACR2	TIMAFLTR	TIMAOUTR	TIMACHPR	TIMARSTR	TIMAEEFR2	TIMAEEFR1	TIMARST2R	TIMASET2R	TIMARST1R	TIMASET1R	TIMADTxR	TIMACMP4	TIMACMP3	TIMACMP2	TIMACMP1	TIMAREP	TIMAPER	TIMACNT	TIMADIER	TIMAICR	TIMACR
	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
0x3E0	HRTIM_ BDTBUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMBEEFR3	TIMBCR2	TIMBFLTR	TIMBOUTR	TIMBCHPR	TIMBRSTR	TIMBEEFR2	TIMBEEFR1	TIMBRST2R	TIMBSET2R	TIMBRST1R	TIMBSET1R	TIMBDTxR	TIMBCMP4	TIMBCMP3	TIMBCMP2	TIMBCMP1	TIMBREP	TIMBPER	TIMBCNT	TIMBDIER	TIMBICR	TIMBCR
	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
0x3E4	HRTIM_ BDTCUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMCEEFR3	TIMCCR2	TIMCFLTR	TIMCOUTR	TIMCCHPR	TIMCRSTR	TIMCEEFR2	TIMCEEFR1	TIMCRST2R	TIMCSET2R	TIMCRST1R	TIMCSET1R	TIMCDTxR	TIMCCMP4	TIMCCMP3	TIMCCMP2	TIMCCMP1	TIMCREP	TIMCPER	TIMCCNT	TIMCDIER	TIMCICR	TIMCCR
	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
0x3E8	HRTIM_ BDTDUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMDEEFR3	TIMDCR2	TIMDFLTR	TIMDOUTR	TIMDCHPR	TIMDRSTR	TIMDEEFR2	TIMDEEFR1	TIMDRST2R	TIMDSET2R	TIMDRST1R	TIMDSET1R	TIMDDTxR	TIMDCMP4	TIMDCMP3	TIMDCMP2	TIMDCMP1	TIMDREP	TIMDPER	TIMDCNT	TIMDDIER	TIMDICR	TIMDCR
	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
0x3EC	HRTIM_ BDTEUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEEEFR3	TIMECR2	TIMEFLTR	TIMEOUTR	TIMECHPR	TIMERSTR	TIMEEEFR2	TIMEEEFR1	TIMERST2R	TIMESET2R	TIMERST1R	TIMESET1R	TIMEDTxR	TIMECMP4	TIMECMP3	TIMECMP2	TIMECMP1	TIMEREP	TIMEPER	TIMECNT	TIMEDIER	TIMEICR	TIMECR
	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
0x3F0	HRTIM_ BDMADR	BDMADR[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
0x3F4	HRTIM_ BDTFUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMFEEFR3	TIMFCR2	TIMFFLTR	TIMFOUTR	TIMFCHPR	TIMFRSTR	TIMFEEFR2	TIMFEEFR1	TIMFRST2R	TIMFSET2R	TIMFRST1R	TIMFSET1R	TIMFDTxR	TIMFCMP4	TIMFCMP3	TIMFCMP2	TIMFCMP1	TIMFREP	TIMFPER	TIMFCNT	TIMFDIER	TIMFICR	TIMFCR
	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

Table 243. HRTIM register map and reset values – common functions (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
0x3F8	HRTIM_ADCER[1]	Res.	ADC10TRG[4:0]					ADC9TRG[4:0]					ADC8TRG[4:0]					Res.	ADC7TRG[4:0]					ADC6TRG[4:0]					ADC5TRG[4:0]				
	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
0x3FC	HRTIM_ADCUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AD10USRC[2:0]			Res.	AD9USRC[2:0]			Res.	AD8USRC[2:0]			Res.	AD7USRC[2:0]			Res.	AD6USRC[2:0]			Res.	AD5USRC[2:0]		
	Reset value	-	-	-	-	-	-	-	-	-	0	0	0	-	0	0	0	-	0	0	0	-	0	0	0	-	0	0	0	-	0	0	0
0x400	HRTIM_ADCPS1	Res.	Res.	Res.	ADC5PSC[4:0]					Res.	ADC4PSC[4:0]					Res.	ADC3PSC[4:0]					Res.	ADC2PSC[4:0]					Res.	ADC1PSC[4:0]				
	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
0x404	HRTIM_ADCPS2	Res.	Res.	Res.	ADC10PSC[4:0]					Res.	ADC9PSC[4:0]					Res.	ADC8PSC[4:0]					Res.	ADC7PSC[4:0]					Res.	ADC6PSC[4:0]				
	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
0x408	HRTIM_FLTINR3	FLT4RSTM	FLT4CRES	FLT4CNT[3:0]				FLT4BLKS	FLT4BLKE	FLT3RSTM	FLT3CRES	FLT3CNT[3:0]				FLT3BLKS	FLT3BLKE	FLT2RSTM	FLT2CRES	FLT2CNT[3:0]				FLT2BLKS	FLT2BLKE	FLT1RSTM	FLT1CRES	FLT1CNT[3:0]				FLT1BLKS	FLT1BLKE
	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
0x40C	HRTIM_FLTINR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT6RSTM	FLT6CRES	FLT6CNT[3:0]				FLT6BLKS	FLT6BLKE	FLT5RSTM	FLT5CRES	FLT5CNT[3:0]				FLT5BLKS	FLT5BLKE
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This register can be preloaded (see *Table 226 on page 908*).

Refer to *Section 2.2 on page 81* for the register boundary addresses.

28 Advanced-control timers (TIM1/TIM8/TIM20)

28.1 TIM1/TIM8/TIM20 introduction

The advanced-control timers (TIM1/TIM8/TIM20) consist 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/TIM8/TIM20) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 28.3.30: Timer synchronization*.

28.2 TIM1/TIM8/TIM20 main features

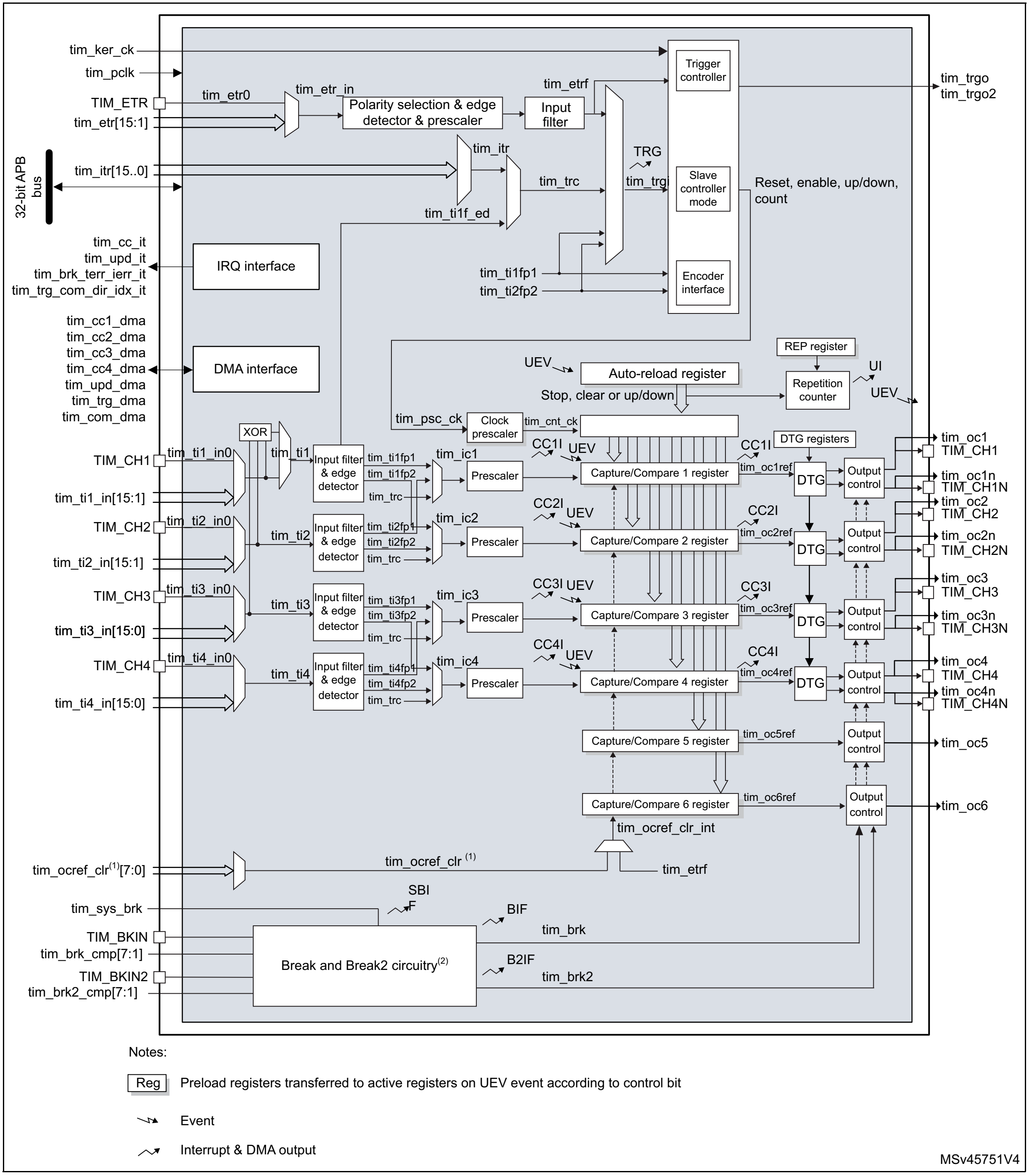
TIM1/TIM8/TIM20 timer features include:

- 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

28.3 TIM1/TIM8/TIM20 functional description

28.3.1 Block diagram

Figure 269. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals*.
2. See *Figure 316: Break and Break2 circuitry overview* for details.

28.3.2 TIM1/TIM8/TIM20 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 244. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input / Output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 245. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 245. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[7:1]	Input	Break input for internal signals
tim_brk2_cmp[7:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trg_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trg_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 246, *Table 247*, *Table 248* and *Table 249* list the sources connected to the tim_ti[1..4] input multiplexers.

Table 246. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti1_in0	TIM1_CH1	TIM8_CH1	TIM20_CH1
tim_ti1_in1	comp1_out	comp1_out	comp1_out
tim_ti1_in2	comp2_out	comp2_out	comp2_out
tim_ti1_in3	comp3_out	comp3_out	comp3_out
tim_ti1_in4	comp4_out	comp4_out	comp4_out
tim_ti1_in[5..15]	Reserved		

Table 247. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti2_in0	TIM1_CH2	TIM8_CH2	TIM20_CH2
tim_ti2_in[1..15]	Reserved		

Table 248. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti3_in0	TIM1_CH3	TIM8_CH3	TIM20_CH3
tim_ti2_in[1..15]	Reserved		

Table 249. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti4_in0	TIM1_CH4	TIM8_CH4	TIM20_CH4
tim_ti4_in[1..15]	Reserved		

Table 250 lists the internal sources connected to the tim_itr input multiplexer.

Table 250. TIMx internal trigger connection

TIMx	TIM1	TIM8	TIM20
tim_itr0	Reserved	tim1_trgo	tim1_trgo
tim_itr1	tim2_trgo	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo	tim5_trgo
tim_itr5	tim8_trgo	Reserved	tim8_trgo
tim_itr6	tim15_trgo	tim15_trgo	tim15_trgo
tim_itr7	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr8	tim17_oc1	tim17_oc1	tim17_oc1
tim_itr9	tim20_trgo	tim20_trgo	Reserved
tim_itr10	hrtim_out_sync2	hrtim_out_sync2	hrtim_out_sync2
tim_itr11..15	Reserved		

Table 251 lists the internal sources connected to the tim_etr input multiplexer.

Table 251. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment		
	TIM1	TIM8	TIM20
tim_etr0	TIM1_ETR	TIM8_ETR	TIM20_ETR
tim_etr1	comp1_out	comp1_out	comp1_out
tim_etr2	comp2_out	comp2_out	comp2_out
tim_etr3	comp3_out	comp3_out	comp3_out
tim_etr4	comp4_out	comp4_out	comp4_out
tim_etr5	comp5_out	comp5_out	comp5_out
tim_etr6	comp6_out	comp6_out	comp6_out
tim_etr7	comp7_out	comp7_out	comp7_out
tim_etr8	adc1_awd1	adc2_awd1	adc3_awd1
tim_etr9	adc1_awd2	adc2_awd2	adc3_awd2
tim_etr10	adc1_awd3	adc2_awd3	adc3_awd3
tim_etr11	adc4_awd1	adc3_awd1	adc5_awd1
tim_etr12	adc4_awd2	adc3_awd2	adc5_awd2
tim_etr13	adc4_awd3	adc3_awd3	adc5_awd3
tim_etr[14..15]	Reserved		

Table 252, *Table 253* and *Table 254* list the sources connected to the tim_brk and tim_brk2inputs.

Table 252. Timer break interconnect

tim_brk inputs	TIM1	TIM8	TIM20
TIM_BKIN	TIM1_BKIN pin	TIM8_BKIN pin	TIM20_BKIN pin
tim_brk_cmp1	comp1_out	comp1_out	comp1_out
tim_brk_cmp2	comp2_out	comp2_out	comp2_out
tim_brk_cmp3	comp3_out	comp3_out	comp3_out
tim_brk_cmp4	comp4_out	comp4_out	comp4_out
tim_brk_cmp5	comp5_out	comp5_out	comp5_out
tim_brk_cmp6	comp6_out	comp6_out	comp6_out
tim_brk_cmp7	comp7_out	comp7_out	comp7_out
tim_brk_cmp8	Reserved		

Table 253. Timer break2 interconnect

tim_brk2 inputs	TIM1	TIM8	TIM20
TIM_BKIN2	TIM1_BKIN2 pin	TIM8_BKIN2 pin	TIM20_BKIN2 pin
tim_brk2_cmp1	comp1_out	comp1_out	comp1_out
tim_brk2_cmp2	comp2_out	comp2_out	comp2_out
tim_brk2_cmp3	comp3_out	comp3_out	comp3_out
tim_brk2_cmp4	comp4_out	comp4_out	comp4_out
tim_brk2_cmp5	comp5_out	comp5_out	comp5_out
tim_brk2_cmp6	comp6_out	comp6_out	comp6_out
tim_brk2_cmp7	comp7_out	comp7_out	comp7_out
tim_brk2_cmp8	Reserved		

Table 254. System break interconnect

tim_sys_brk inputs	TIM1 / TIM8 / TIM20	Enable bit in SYSCFG_CFGR2 register
tim_sys_brk0	Cortex®-M4 with FPU LOCKUP	CLL
tim_sys_brk1	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk2	SRAM parity error	SPL
tim_sys_brk3	Flash double ECC error	ECCL
tim_sys_brk4	Clock Security System (CSS)	None (always enabled)

Table 255 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 255. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment		
	TIM1	TIM8	TIM20
tim_ocref_clr0	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out
tim_ocref_clr2	comp3_out	comp3_out	comp3_out
tim_ocref_clr3	comp4_out	comp4_out	comp4_out
tim_ocref_clr4	comp5_out	comp5_out	comp5_out
tim_ocref_clr5	comp6_out	comp6_out	comp6_out
tim_ocref_clr6	comp7_out	comp7_out	comp7_out
tim_ocref_clr7	Reserved		

28.3.3 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, 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 tim_cnt_ck, 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: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides 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 270 and *Figure 271* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 270. Counter timing diagram with prescaler division change from 1 to 2

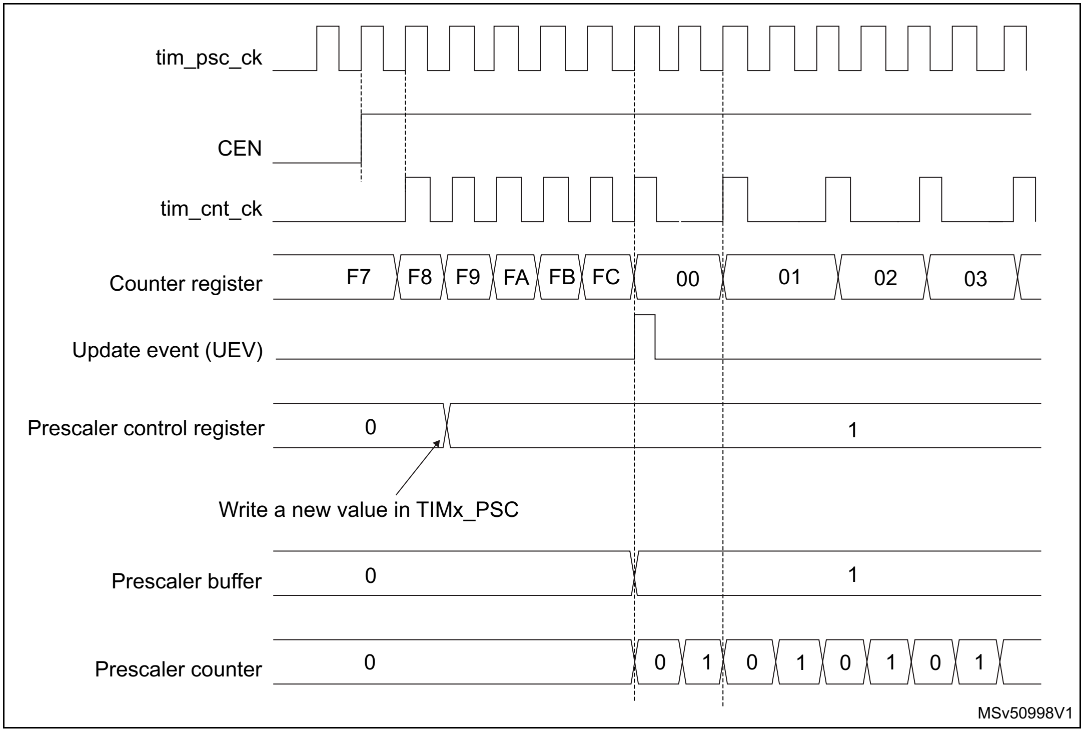
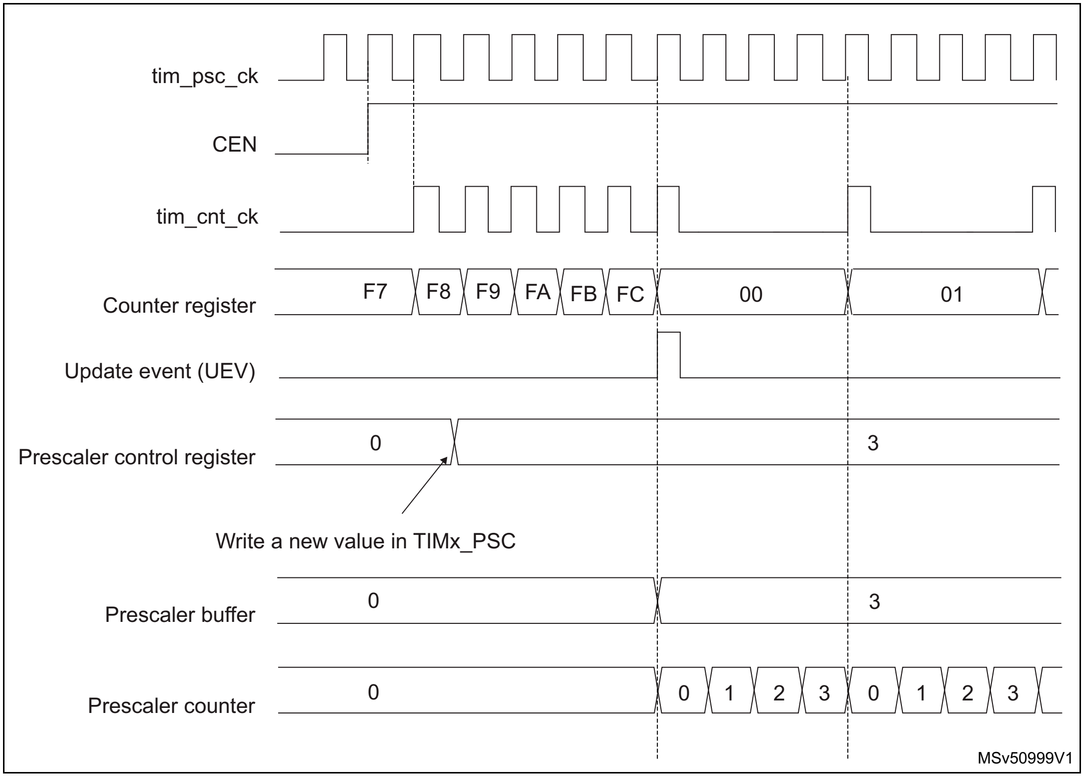


Figure 271. Counter timing diagram with prescaler division change from 1 to 4



28.3.4 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 272. Counter timing diagram, internal clock divided by 1

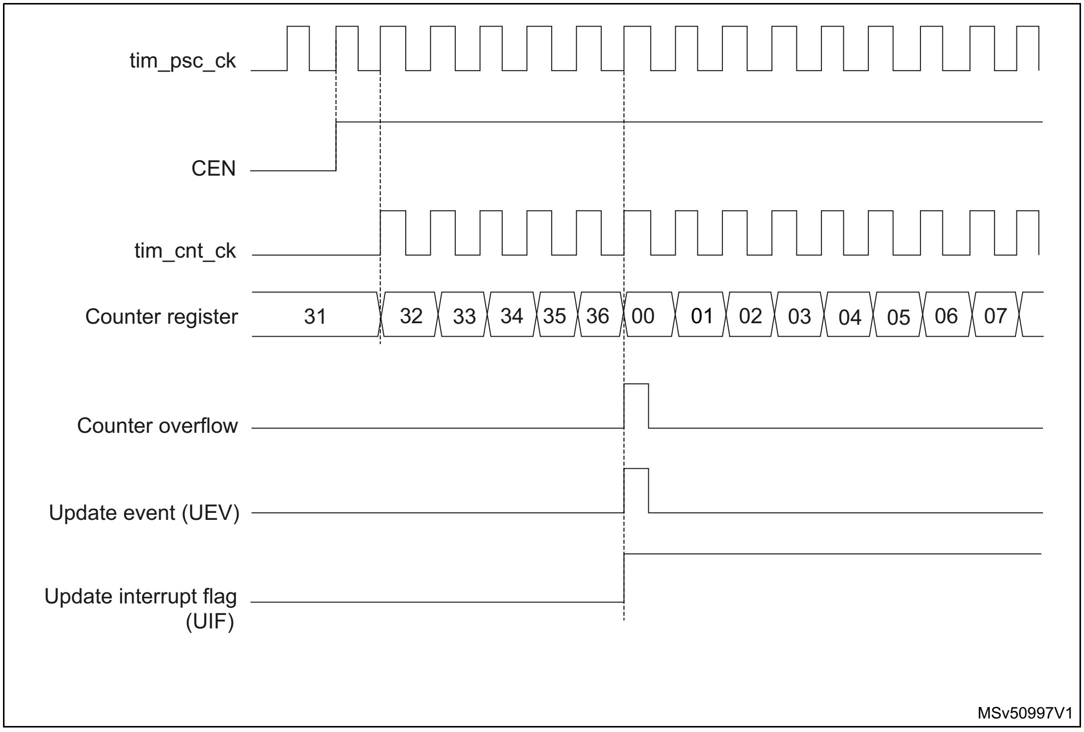


Figure 273. Counter timing diagram, internal clock divided by 2

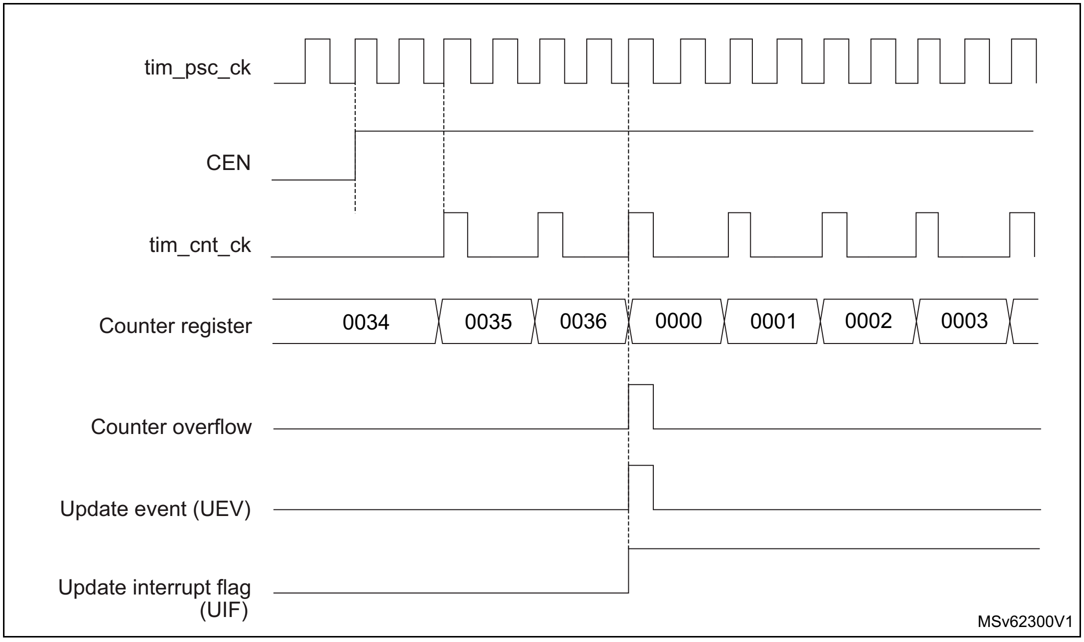


Figure 274. Counter timing diagram, internal clock divided by 4

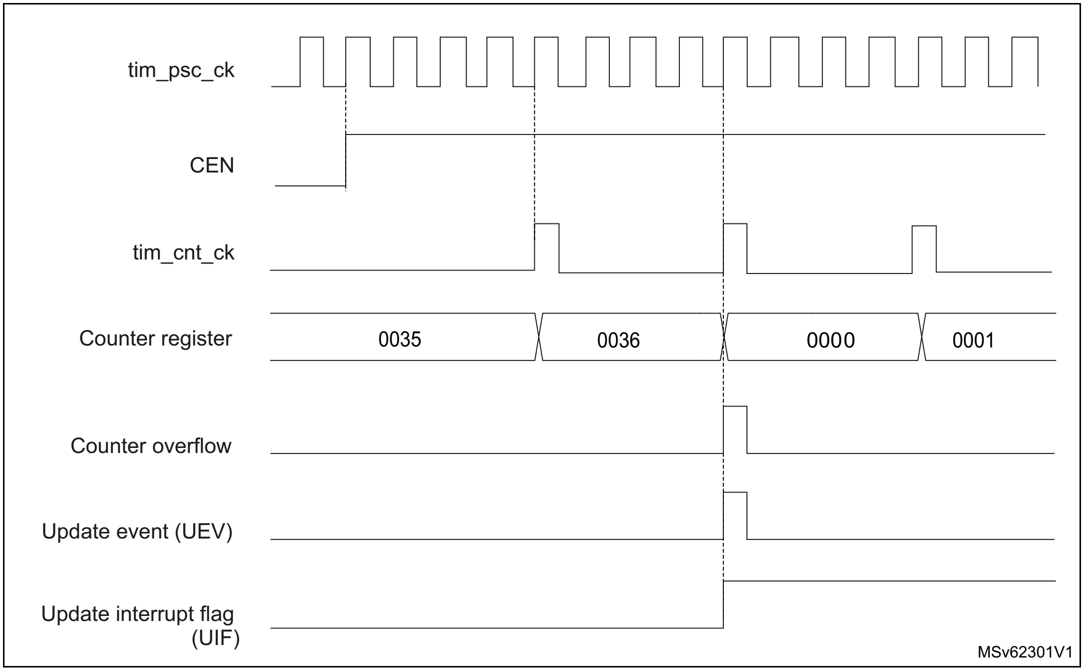


Figure 275. Counter timing diagram, internal clock divided by N

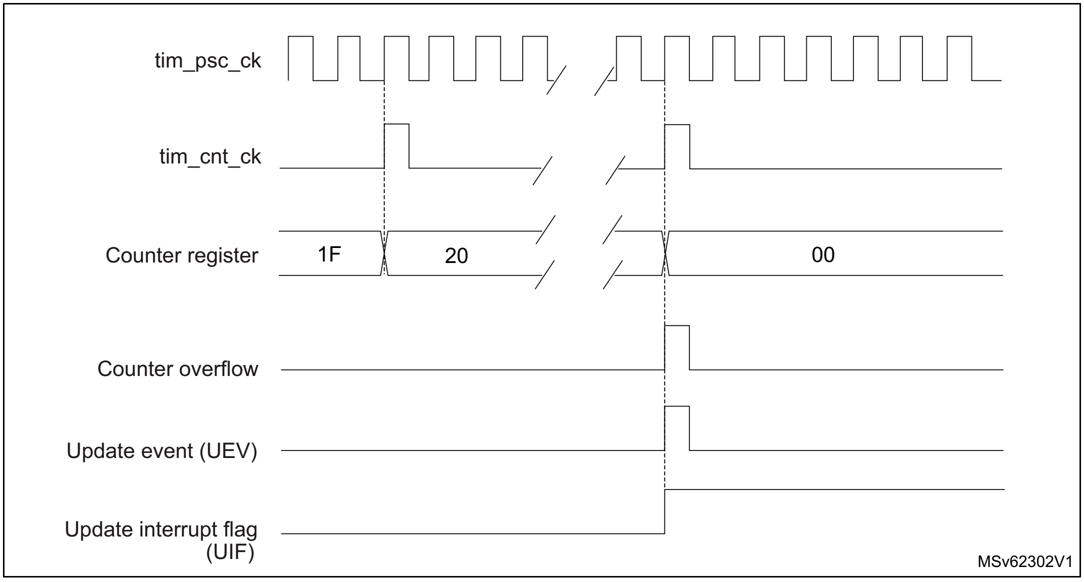


Figure 276. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

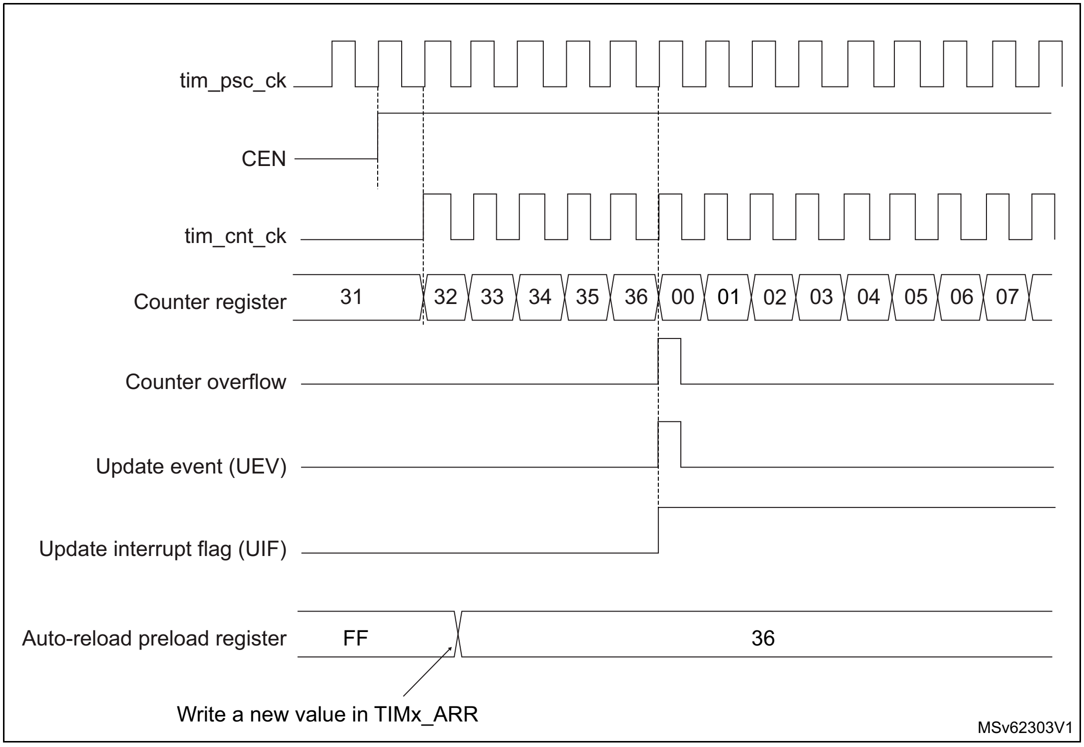
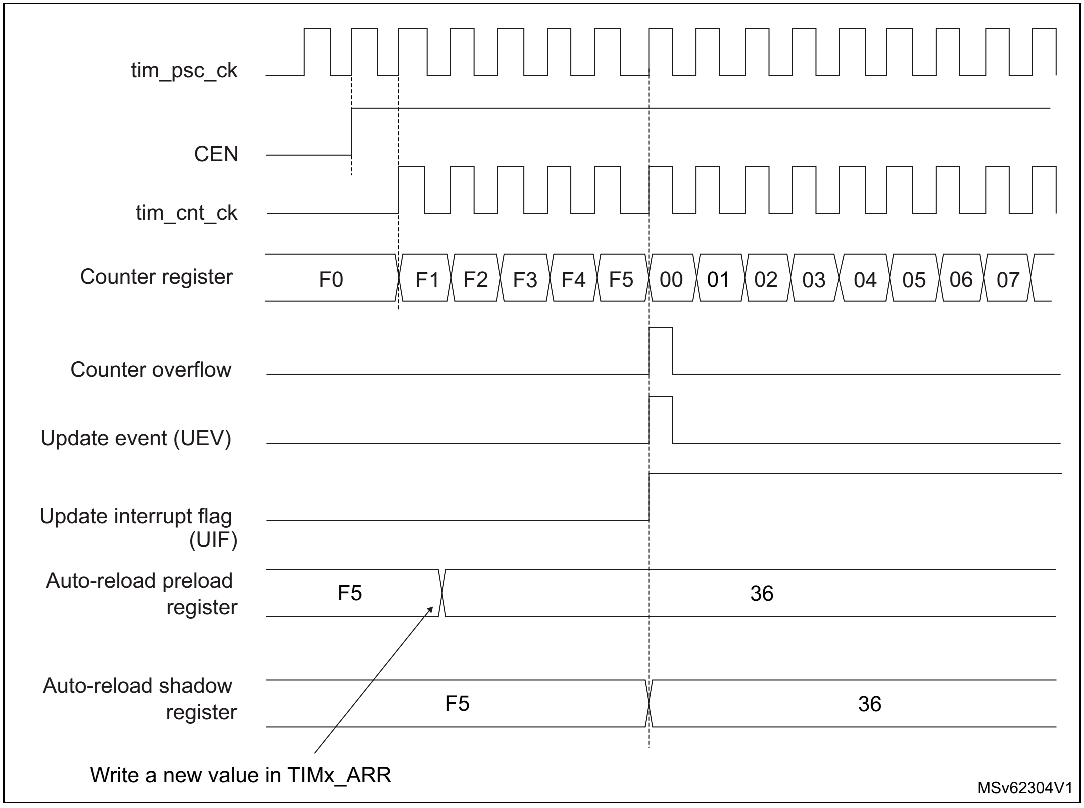


Figure 277. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



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 278. Counter timing diagram, internal clock divided by 1

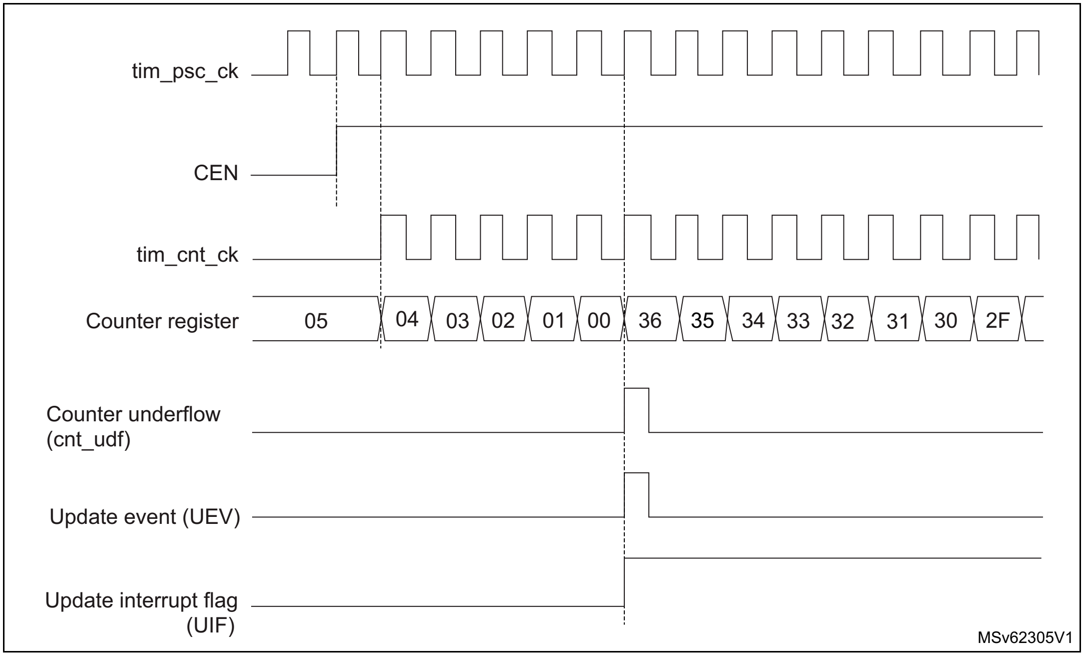


Figure 279. Counter timing diagram, internal clock divided by 2

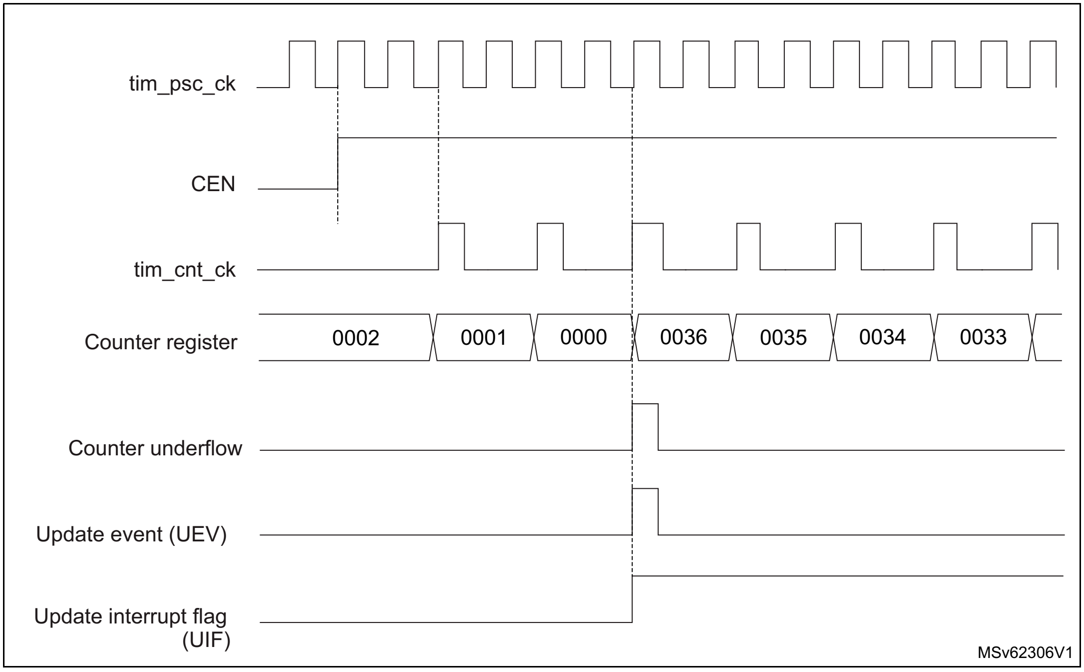


Figure 280. Counter timing diagram, internal clock divided by 4

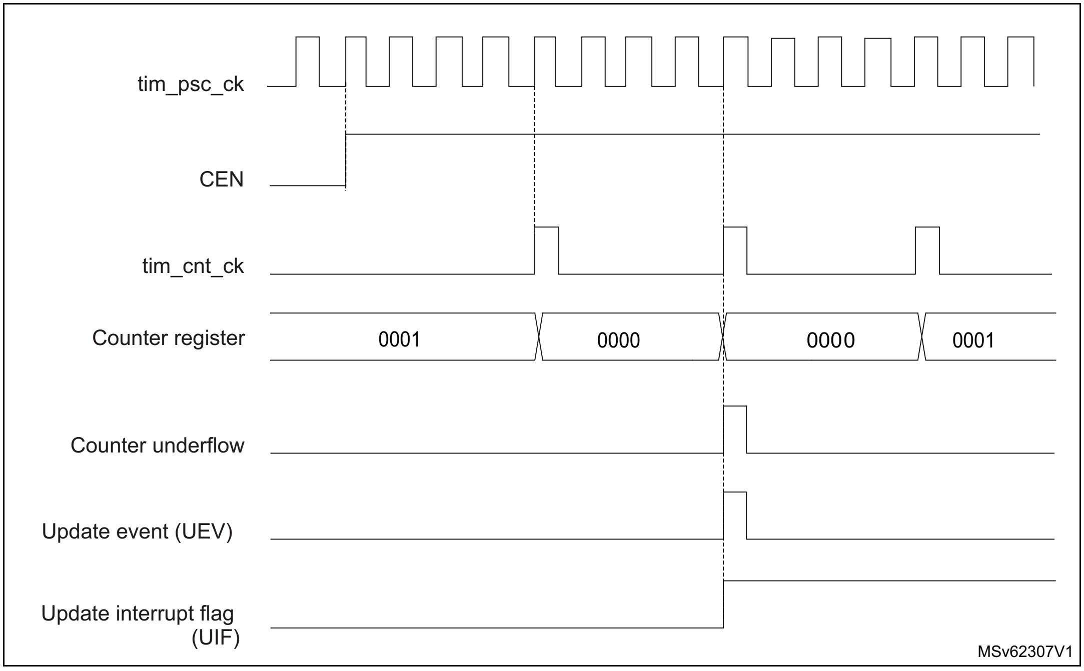


Figure 281. Counter timing diagram, internal clock divided by N

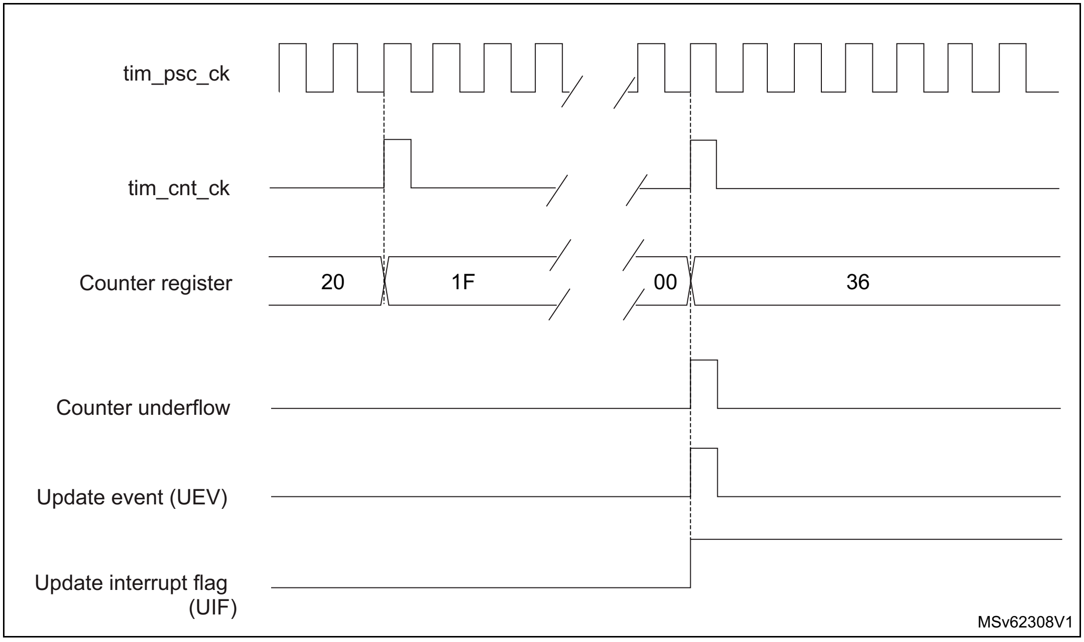
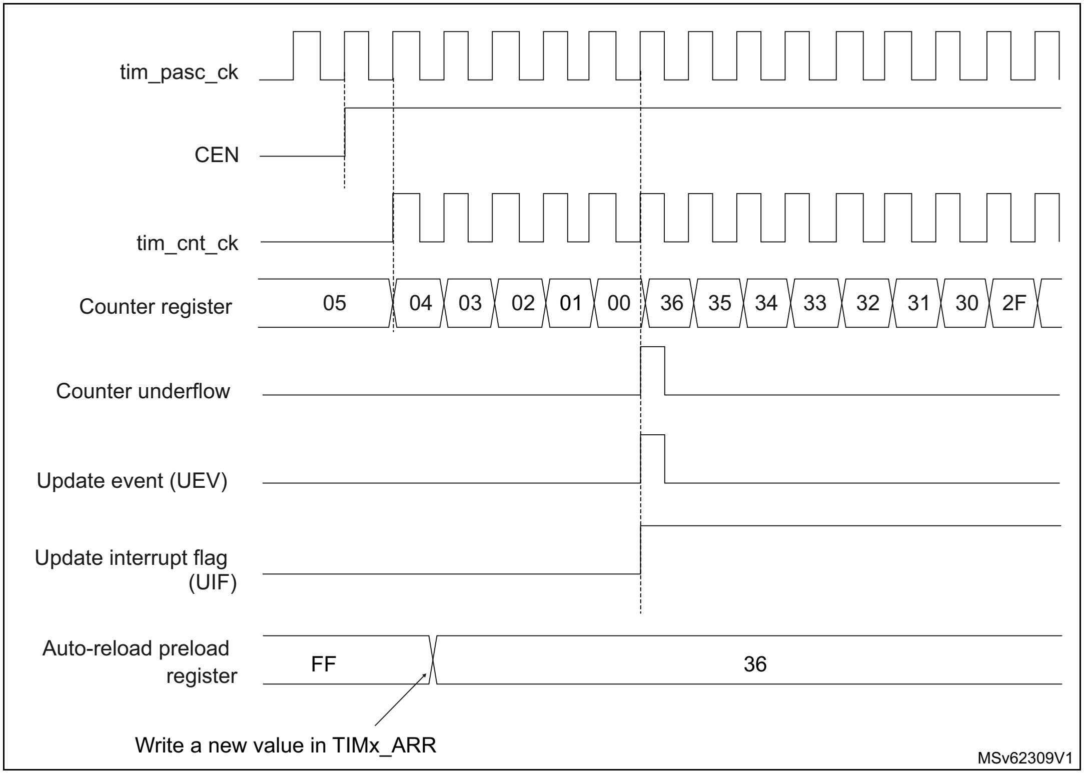


Figure 282. Counter timing diagram, update event when repetition counter is not used



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.

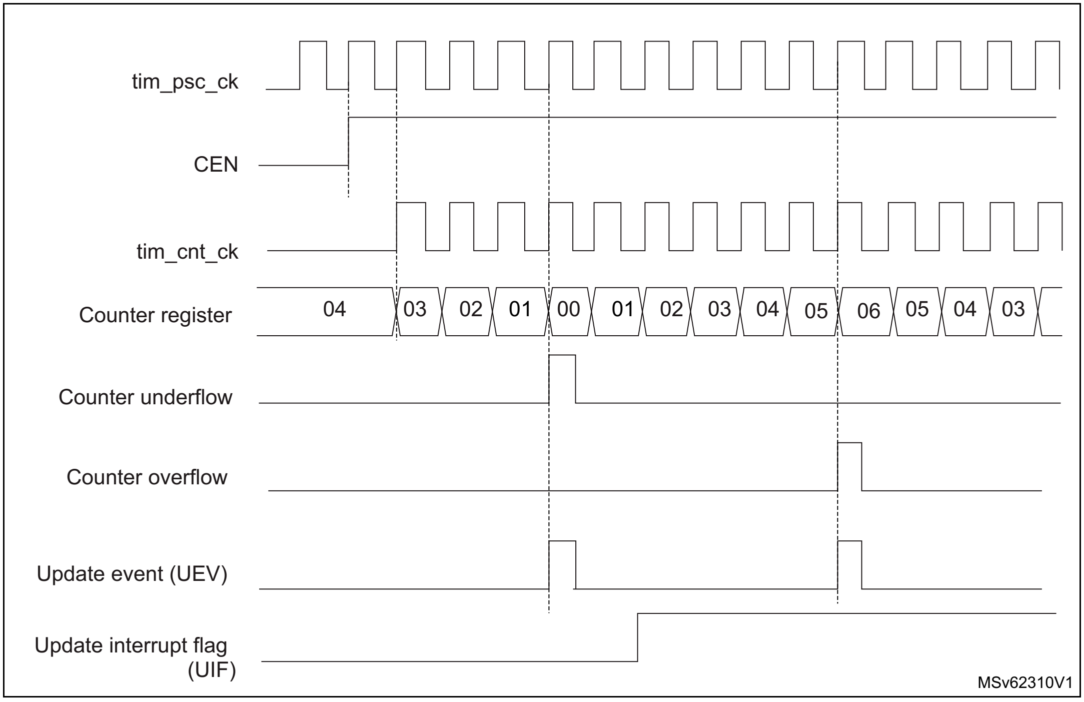
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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 283. 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 28.6: TIM1/TIM8/TIM20 registers*).

Figure 284. Counter timing diagram, internal clock divided by 2

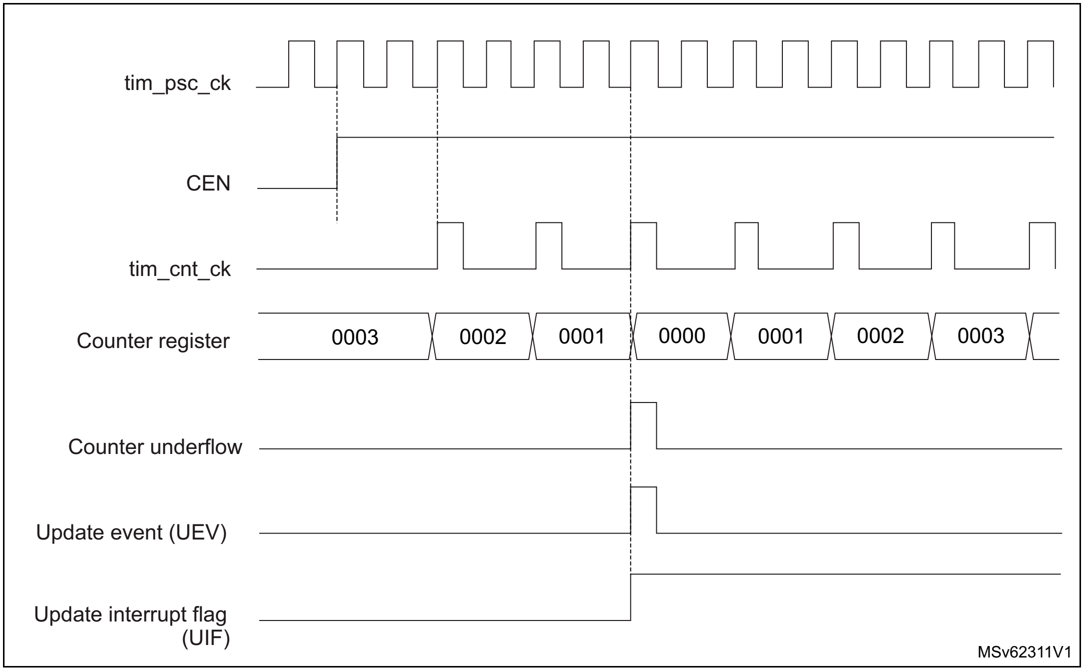


Figure 285. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

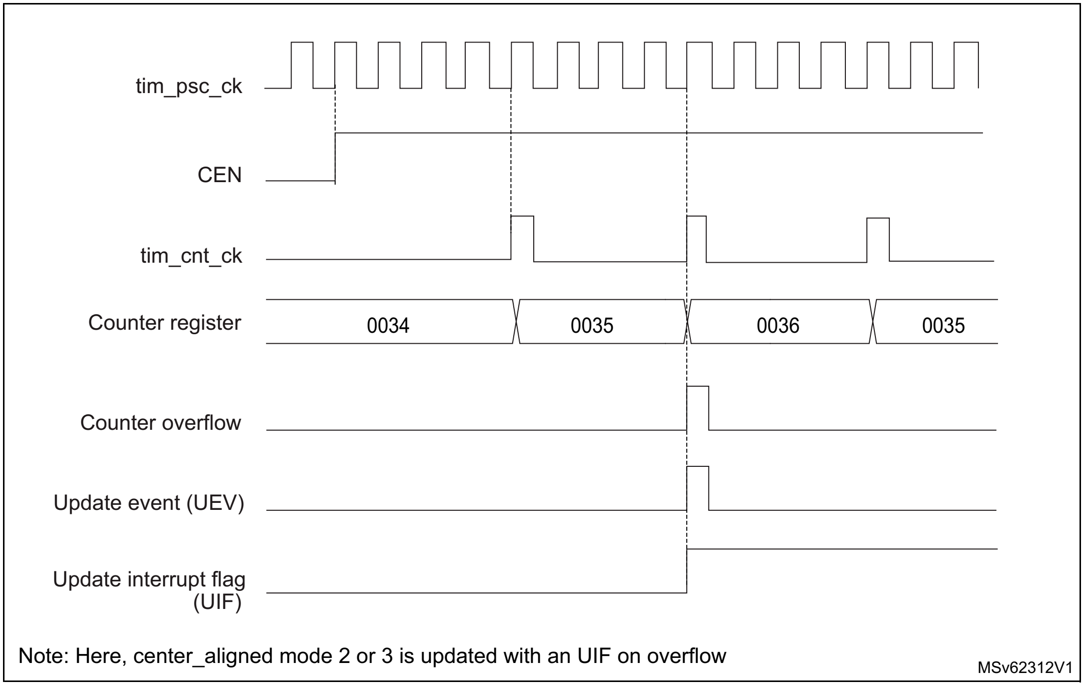

tim_psc_ck
CEN
tim_cnt_ck
Counter register
0034
0035
0036
0035
Counter overflow
Update event (UEV)
Update interrupt flag (UIF)
Note: Here, center_aligned mode 2 or 3 is updated with an UIF on overflow
MSv62312V1

Figure 286. Counter timing diagram, internal clock divided by N

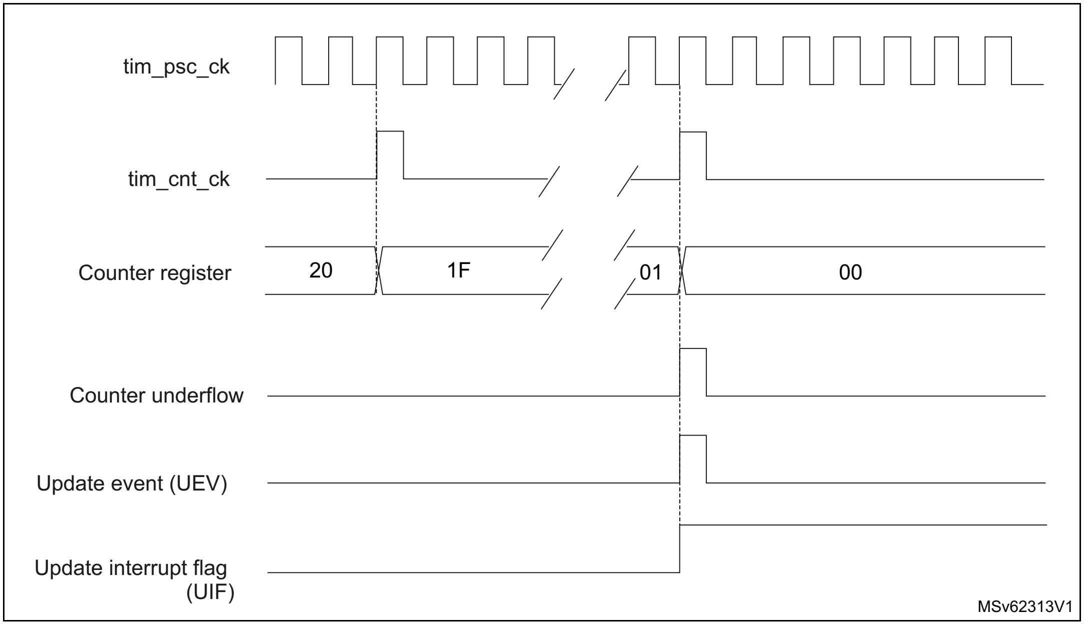


Figure 287. Counter timing diagram, update event with ARPE=1 (counter underflow)

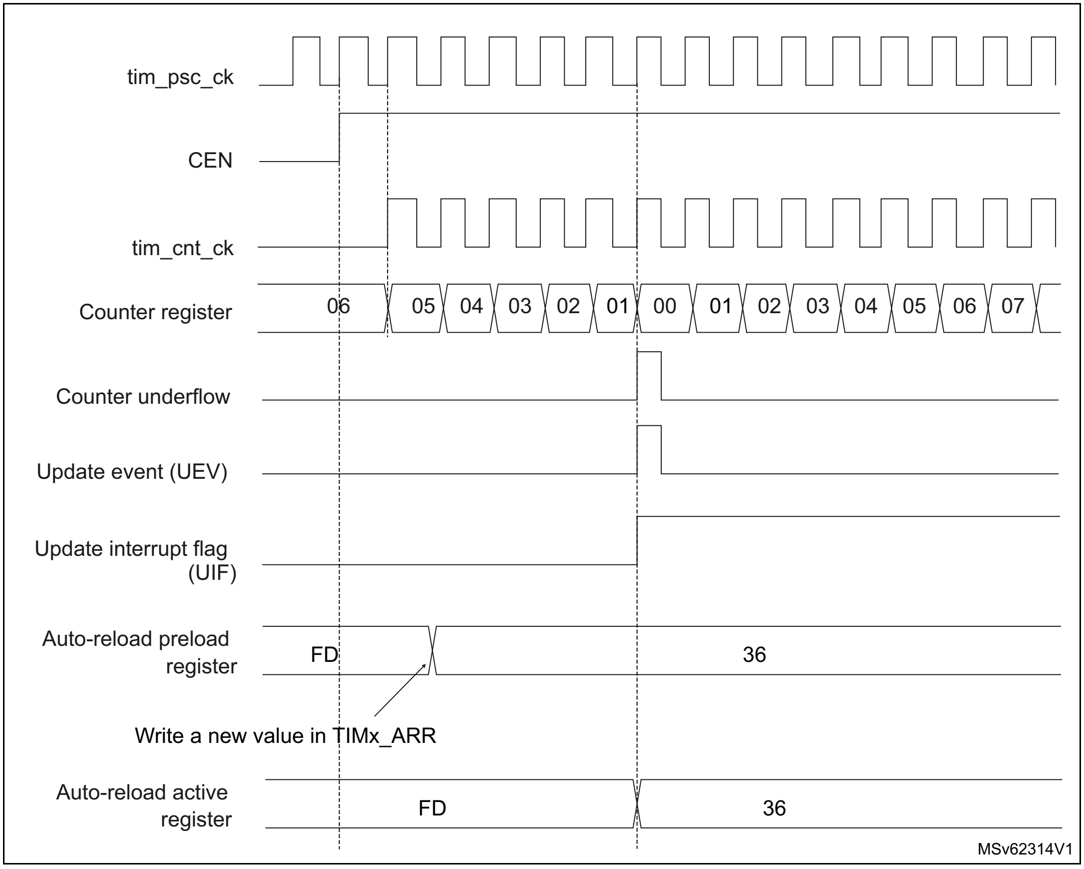
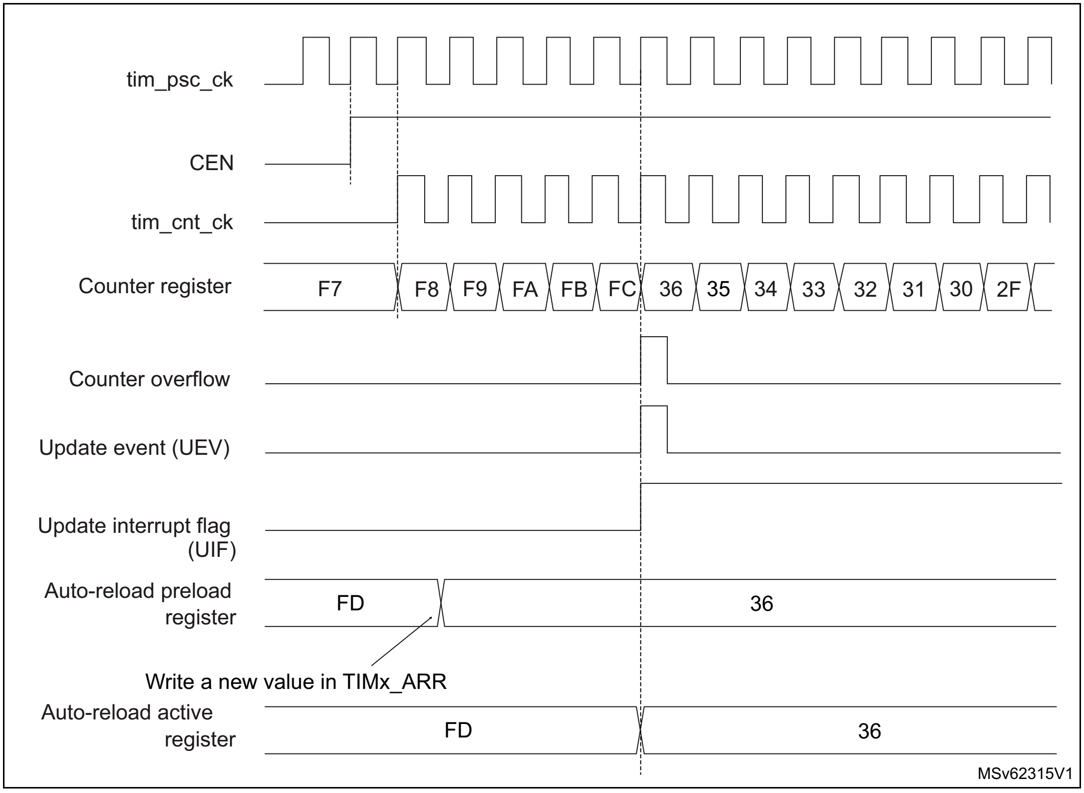


Figure 288. Counter timing diagram, Update event with ARPE=1 (counter overflow)



28.3.5 Repetition counter

Section 28.3.3: 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:

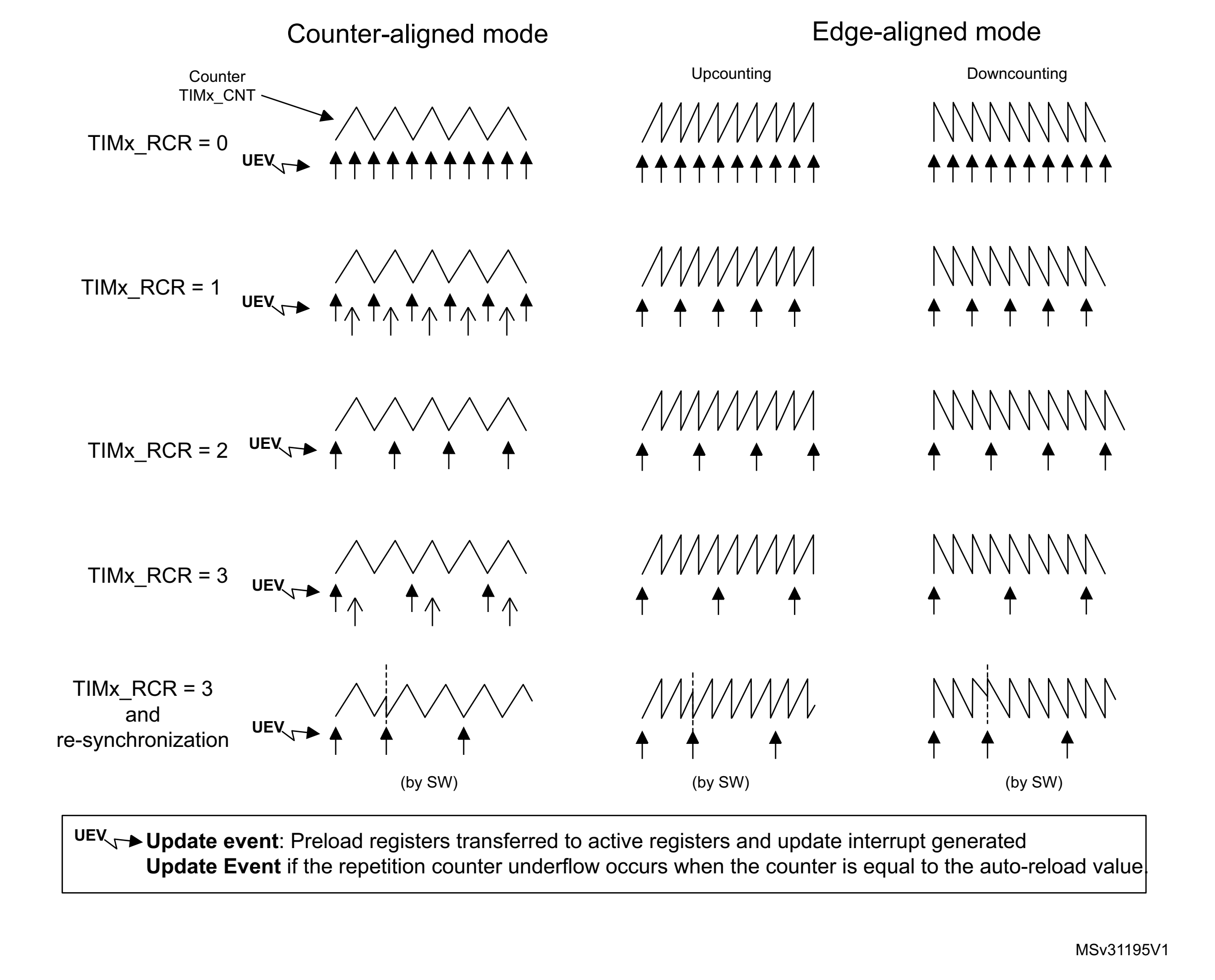
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice 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 289*). 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 289. Update rate examples depending on mode and TIMx_RCR register settings



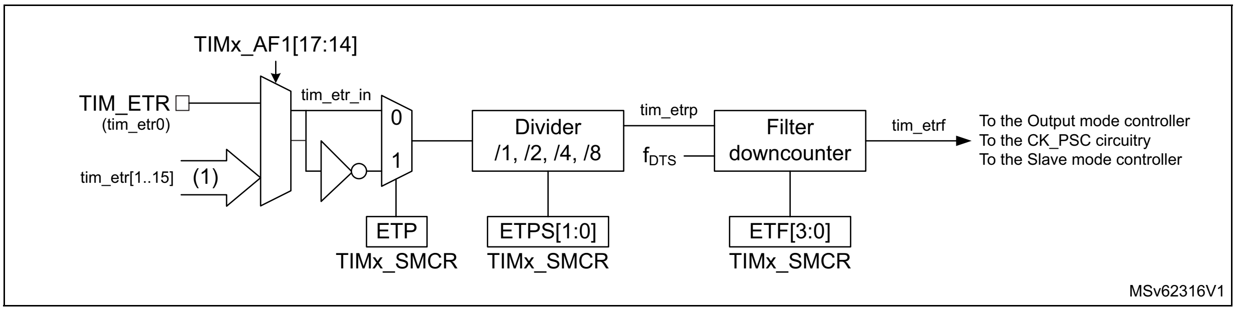
28.3.6 External trigger input

The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 28.3.7*)
- trigger for the slave mode (see *Section 28.3.30*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 28.3.9*)

Figure 290 below describes the tim_etr_in 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. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 290. External trigger input block



The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bitfield in the TIMx_AF1 register.

Refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for the list of sources connected to the etr_in input in the product.

28.3.7 Clock selection

The counter clock can be provided by the following clock sources:

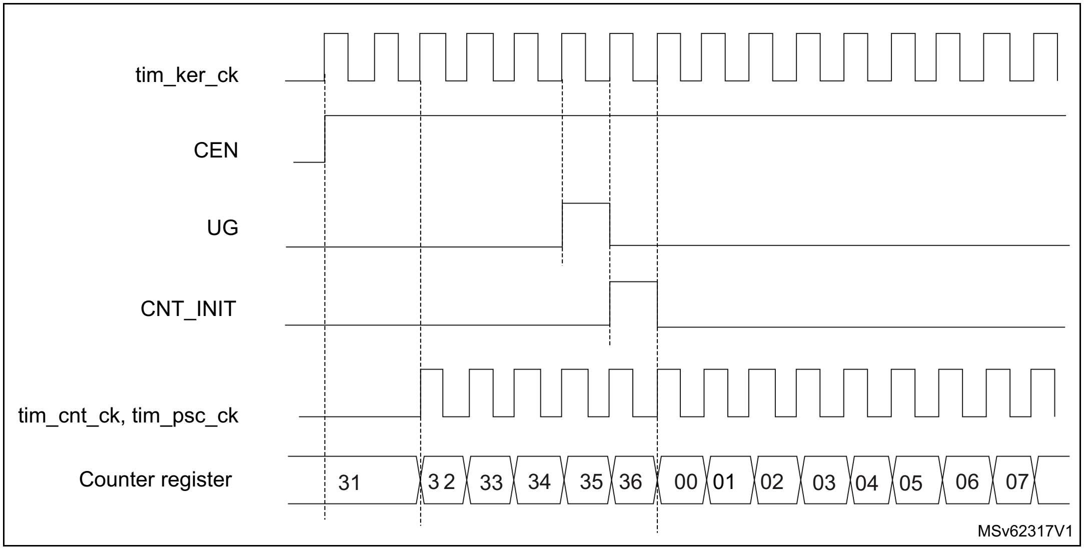
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

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 tim_ker_ck.

Figure 291 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

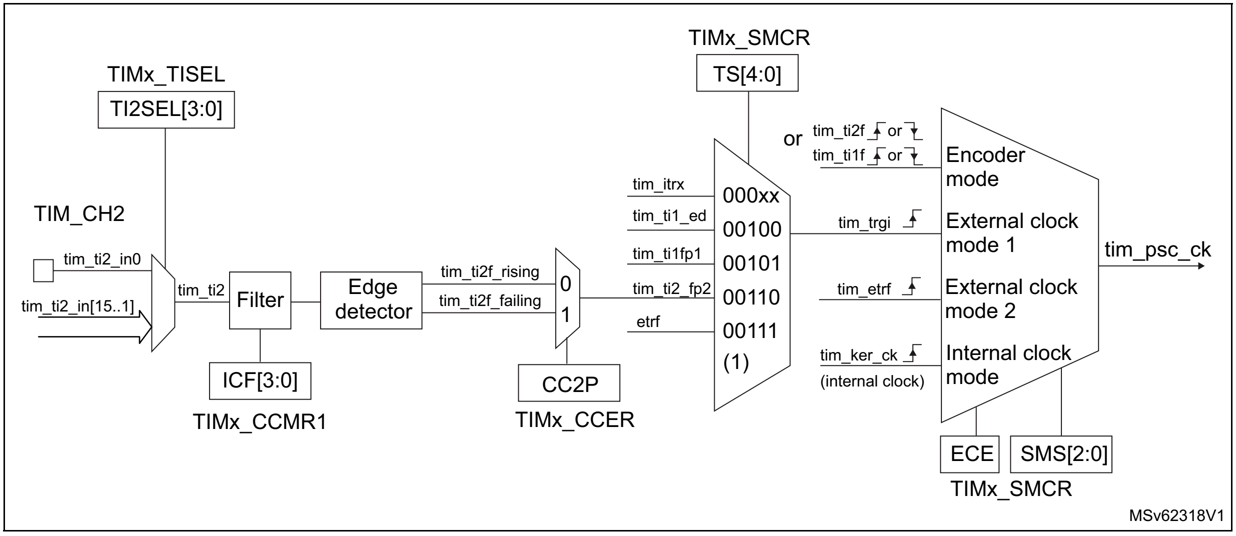
Figure 291. Control circuit in normal mode, internal clock divided by 1



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 292. tim_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 tim_ti2 input, use the following procedure:

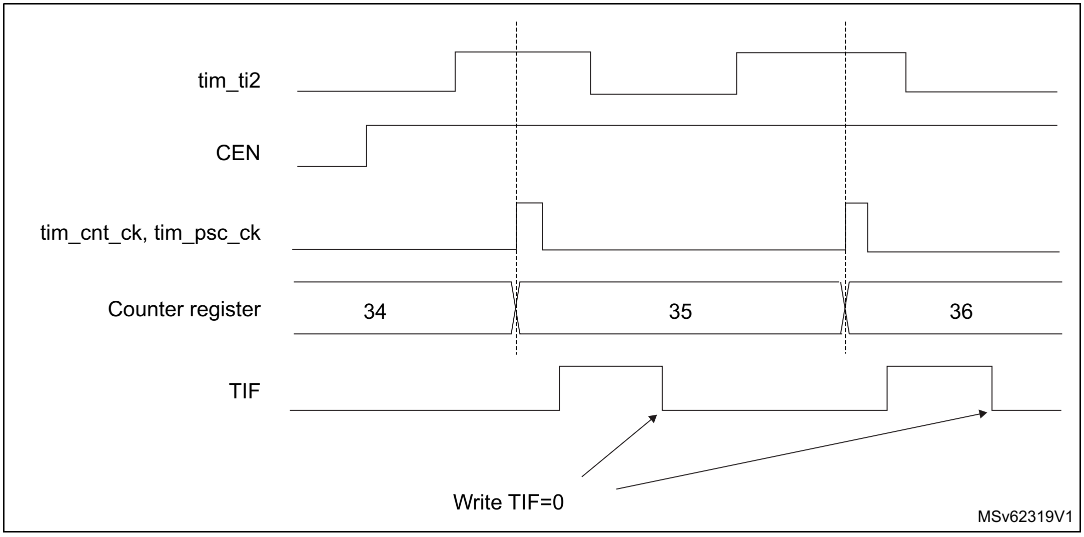
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 293. Control circuit in external clock mode 1



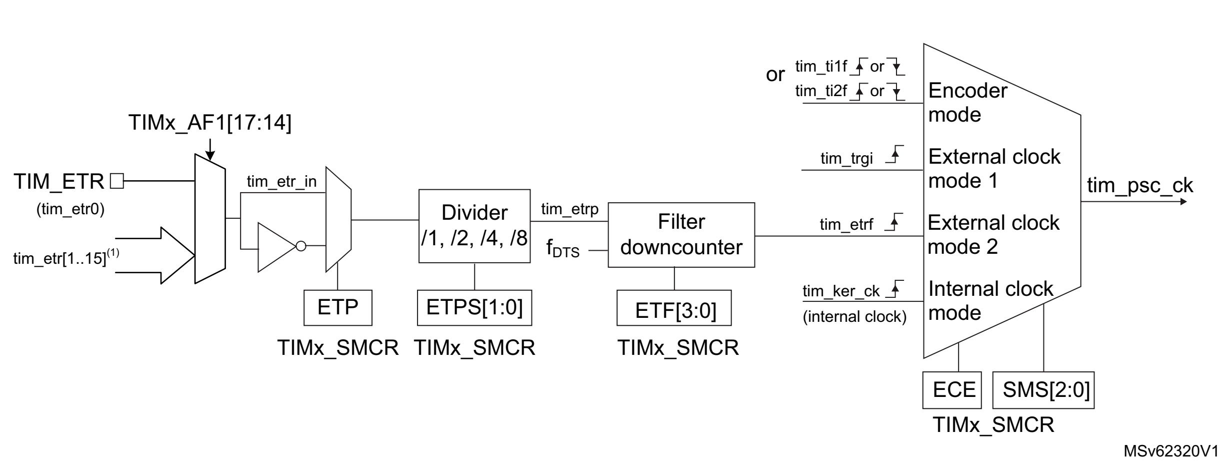
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 294* gives an overview of the external trigger input block.

Figure 294. External trigger input block



1. Refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals*.

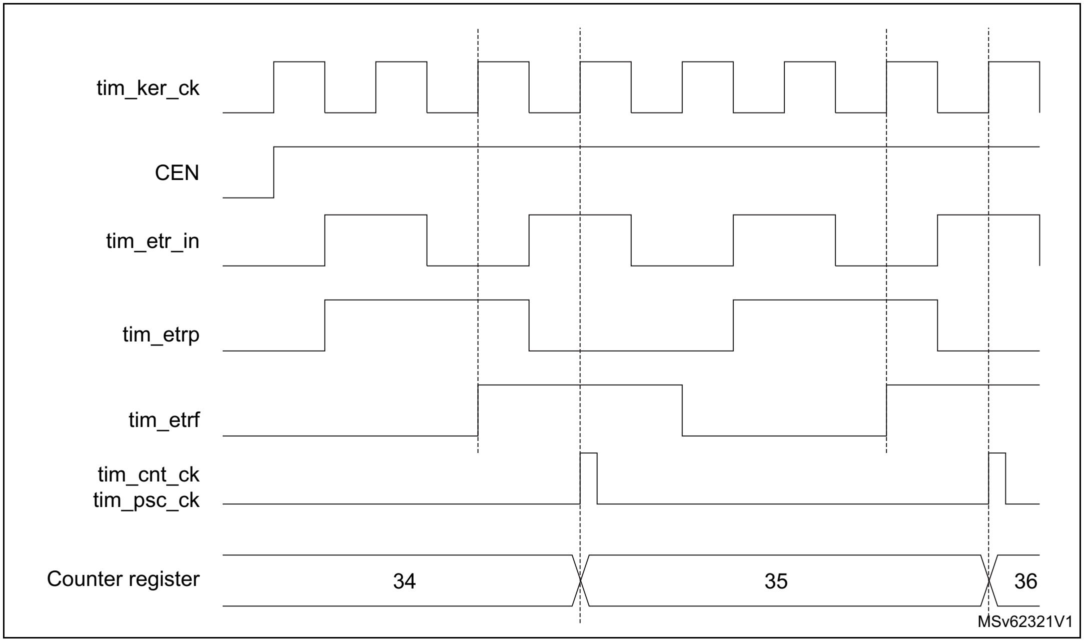
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

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 tim_etr_in input 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 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.

Figure 295. Control circuit in external clock mode 2



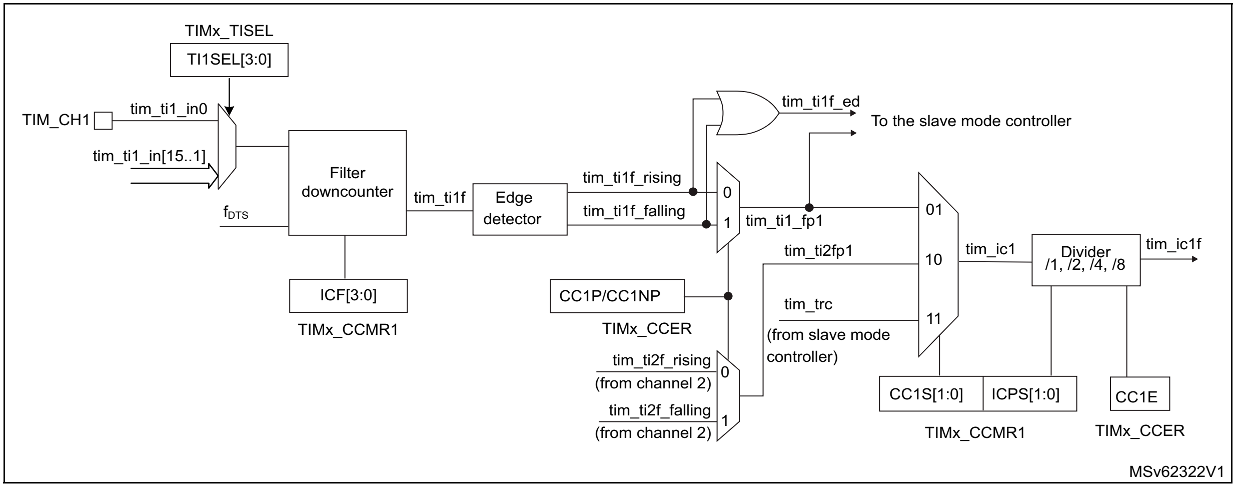
28.3.8 Capture/compare channels

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 296 to *Figure 299* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) 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 296. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 297. Capture/compare channel 1 main circuit

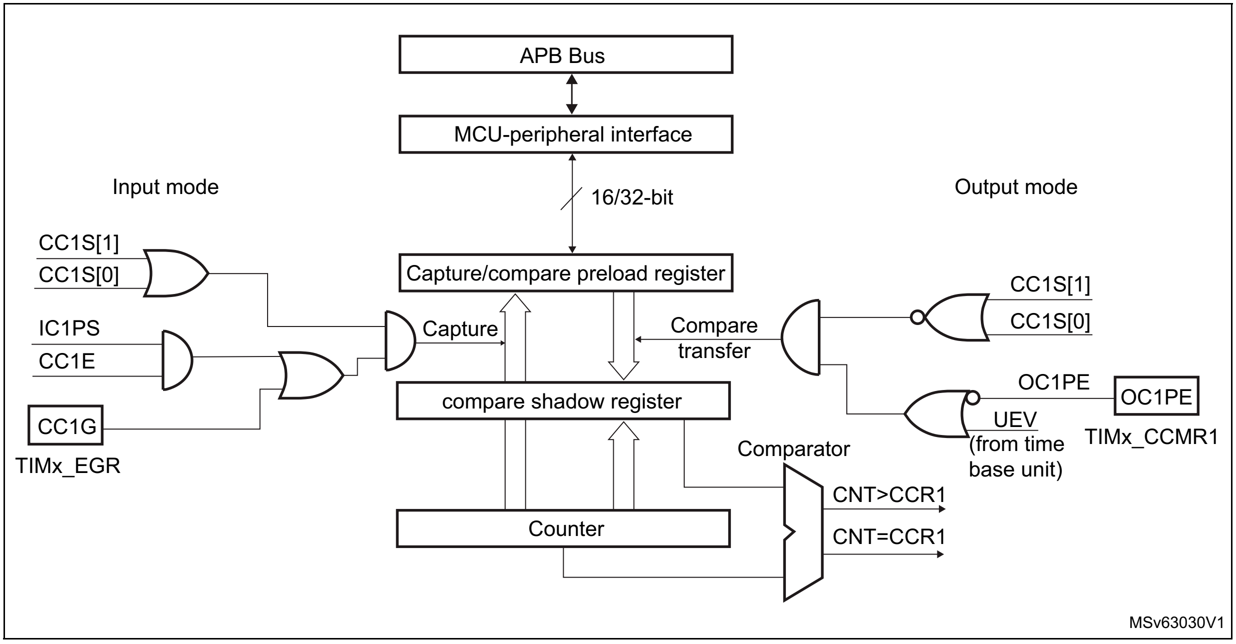
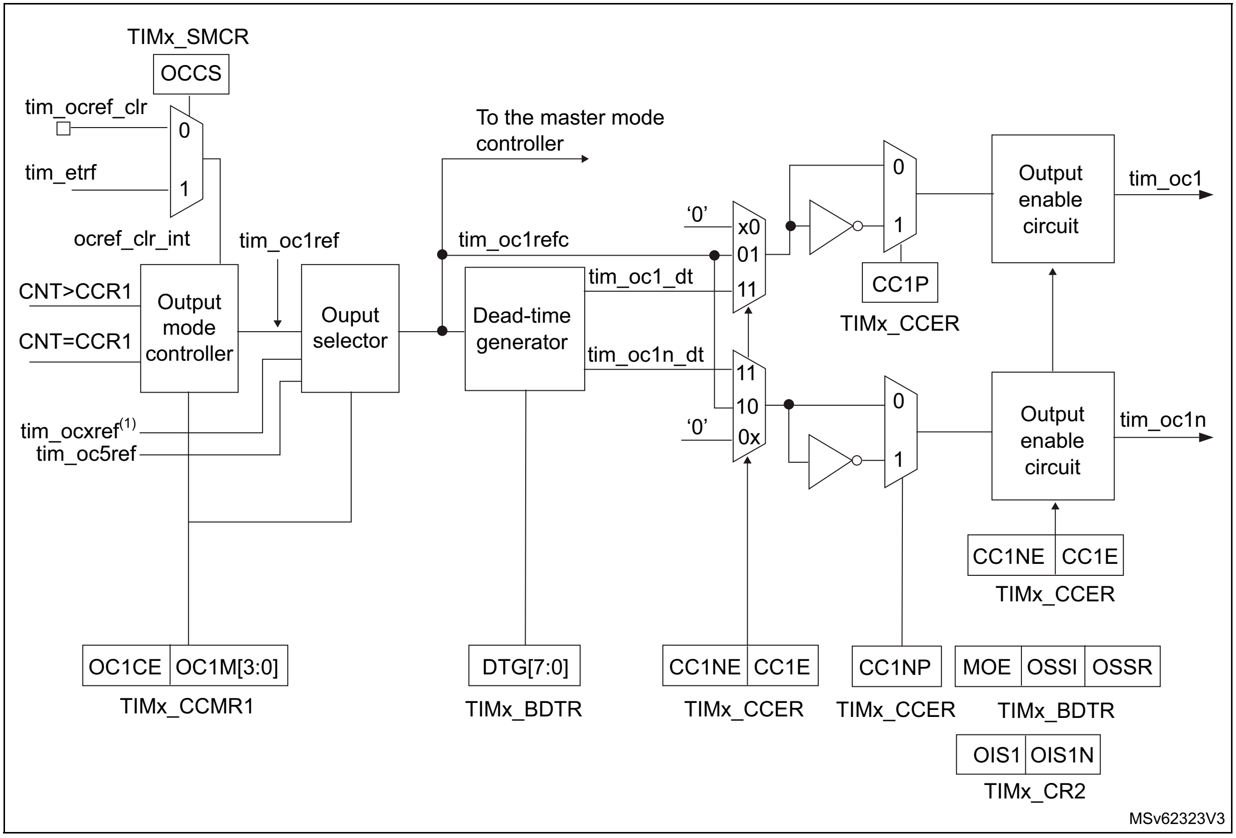
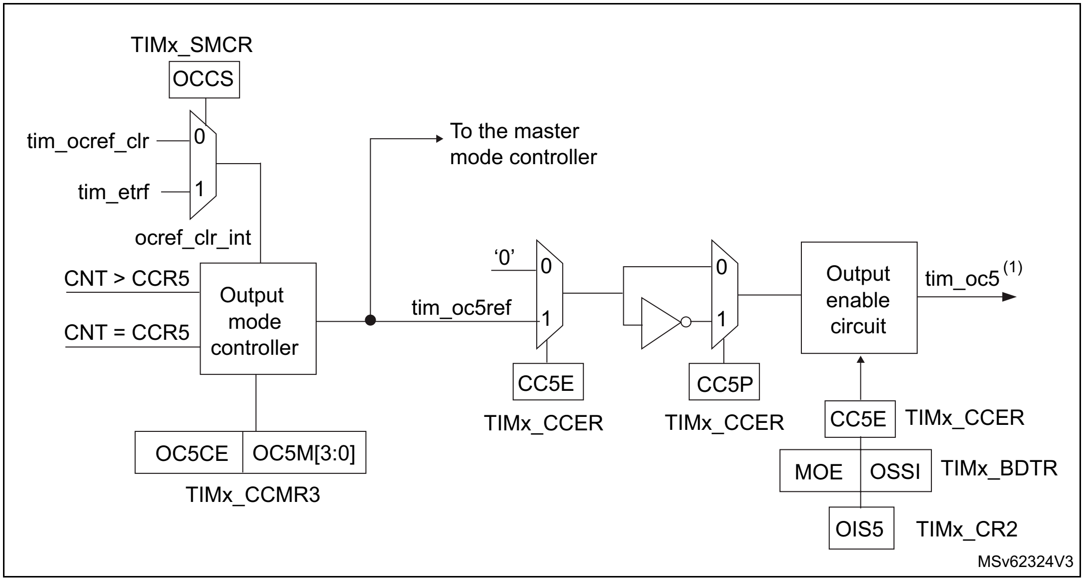


Figure 298. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 299. 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.

28.3.9 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 tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the tim_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.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_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 tim_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.

- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- 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).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- 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.

28.3.10 PWM input mode

This mode allows to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges

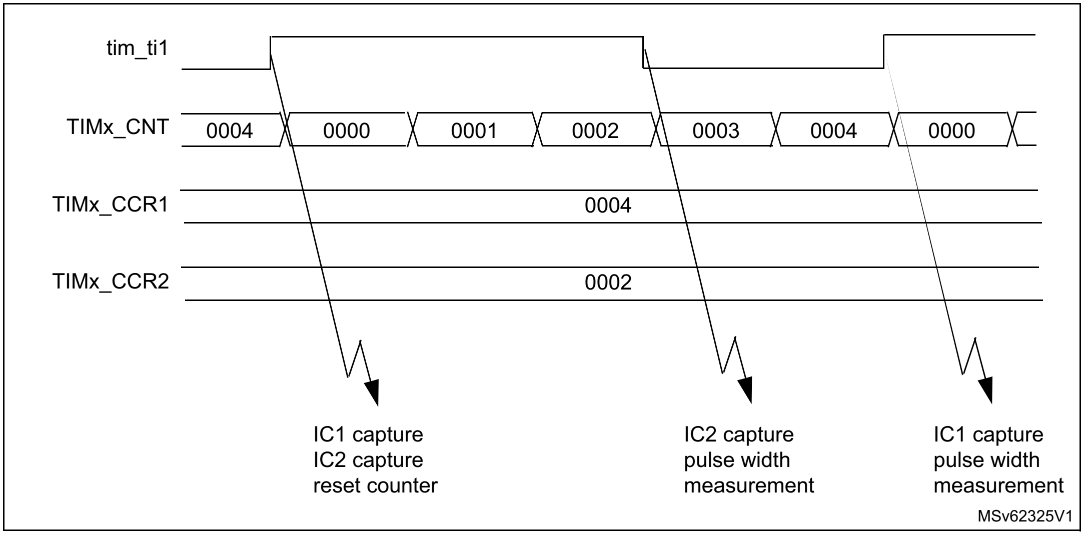
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP=’10’ (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 300. PWM input mode timing



28.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_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 (tim_ocxref/tim_ocx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_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.

28.3.12 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 microcontroller (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.

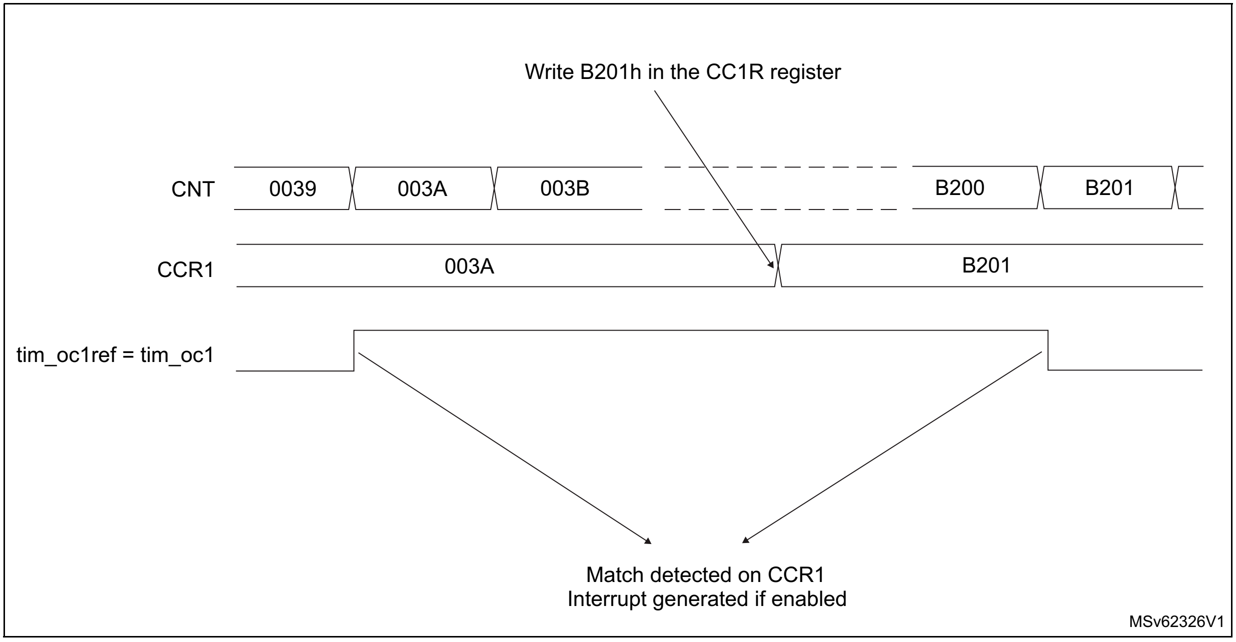
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

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 tim_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 301*.

Figure 301. Output compare mode, toggle on tim_oc1



28.3.13 PWM mode

Pulse Width Modulation mode allows 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 tim_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.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_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.

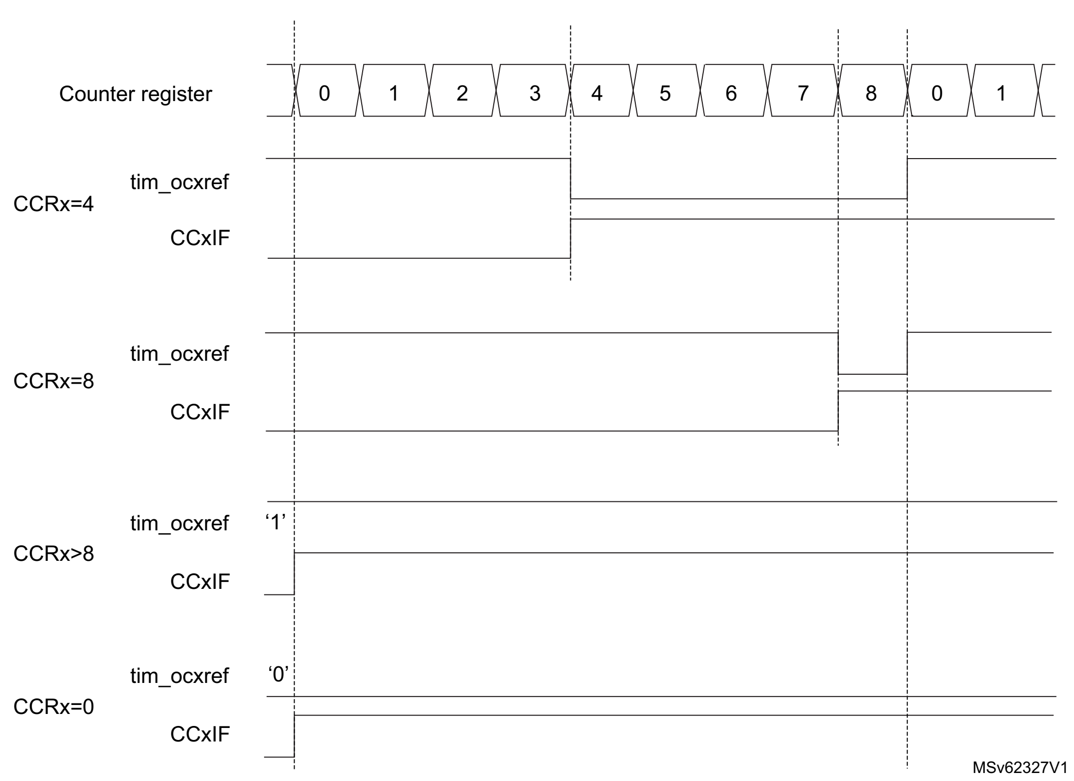
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 1095*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_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 tim_ocxref is held at ‘1’. If the compare value is 0 then tim_ocxref is held at ‘0’. *Figure 302* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 302. 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 1099*

  In PWM mode 1, the reference signal tim_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 tim_ocxref is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

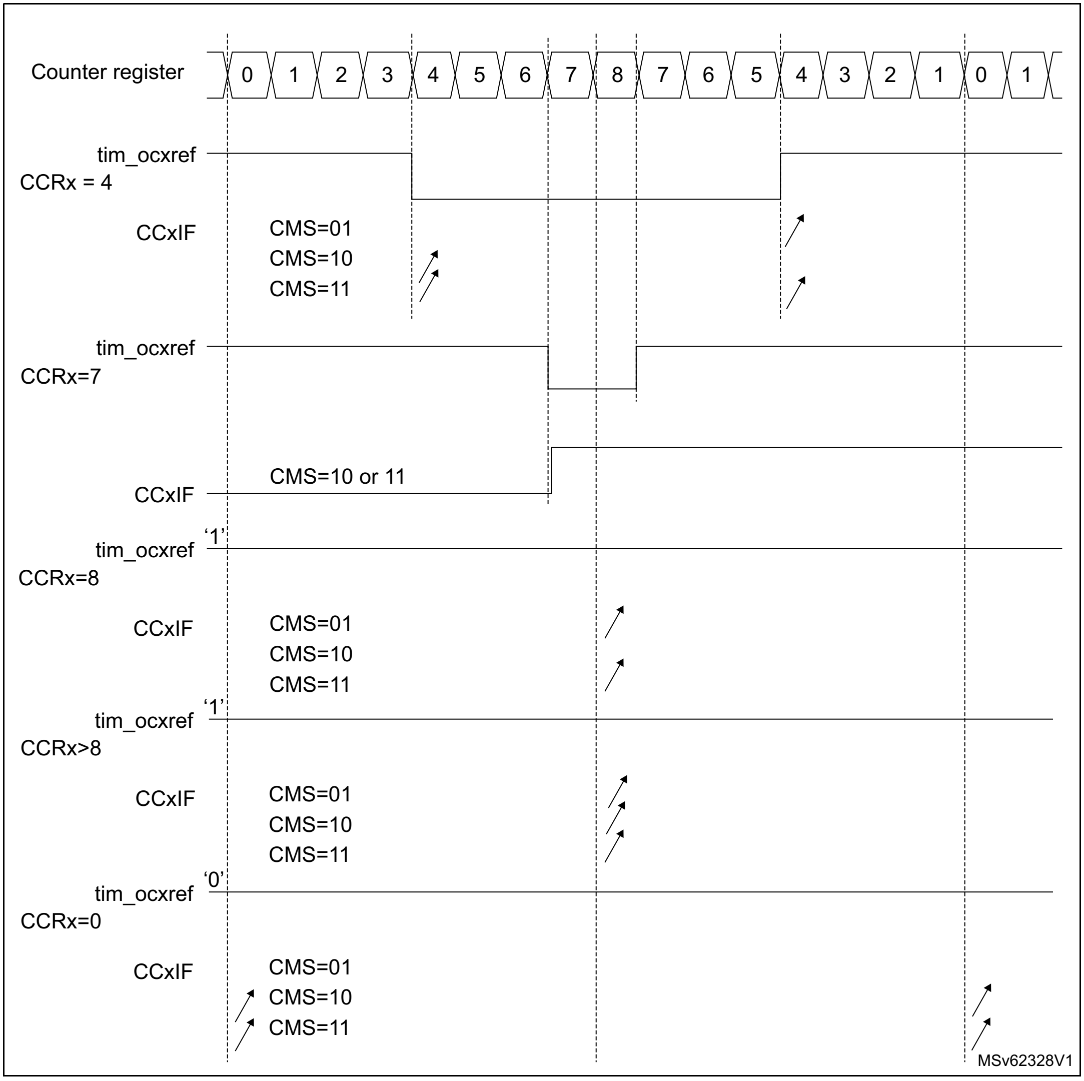
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending 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 1102*.

Figure 303 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 303. 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.

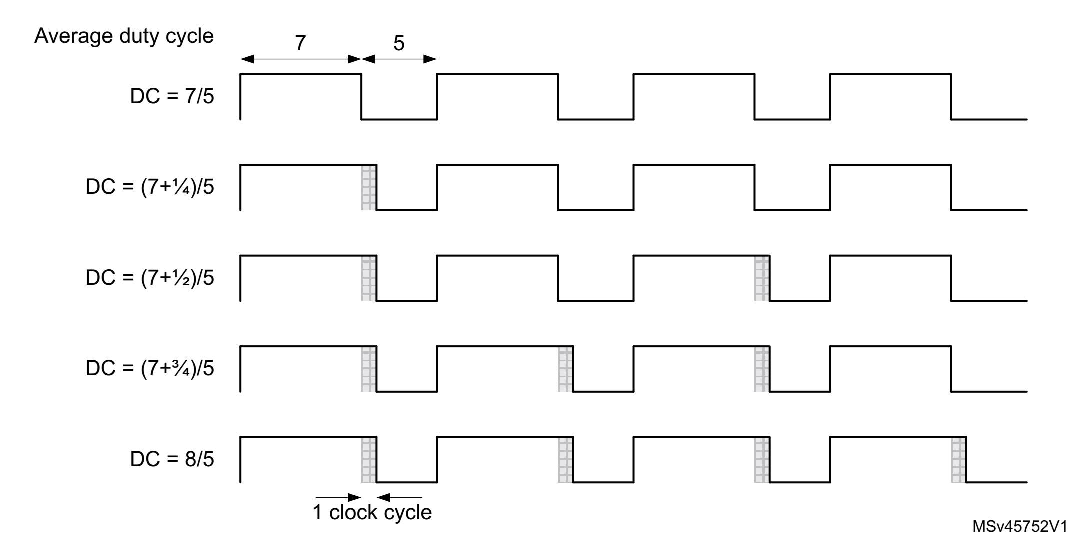
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting 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.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 304* below presents the dithering principle applied to 4 consecutive PWM cycles.

Figure 304. Dithering principle



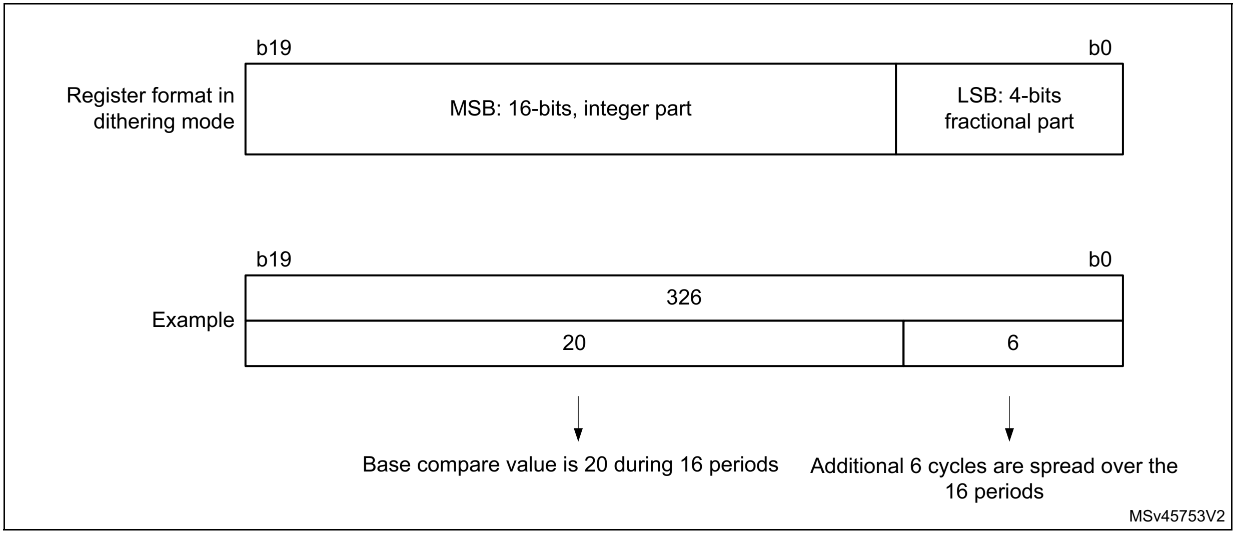
When the dithering mode is enabled, the register coding is changed as follows (see *Figure 305* for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value

Note: The ARR and CCR values will be updated automatically if the DITHEN bit is set / reset (for instance, if ARR= 0x05 with DITHEN=0, it will be updated to ARR = 0x50 with DITHEN=1). The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset

2. *The ARR[3:0] bits must be reset*
3. *The DITHEN bit must be reset*
4. *The CCIF flags must be cleared*
5. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 305. Data format and register coding in dithering mode


Register format in dithering mode
b19
b0
MSB: 16-bits, integer part
LSB: 4-bits fractional part
Example
b19
b0
326
20
6
Base compare value is 20 during 16 periods
Additional 6 cycles are spread over the 16 periods
MSv45753V2

The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \Rightarrow F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{\text{Max}_{\text{Resolution}}}$$

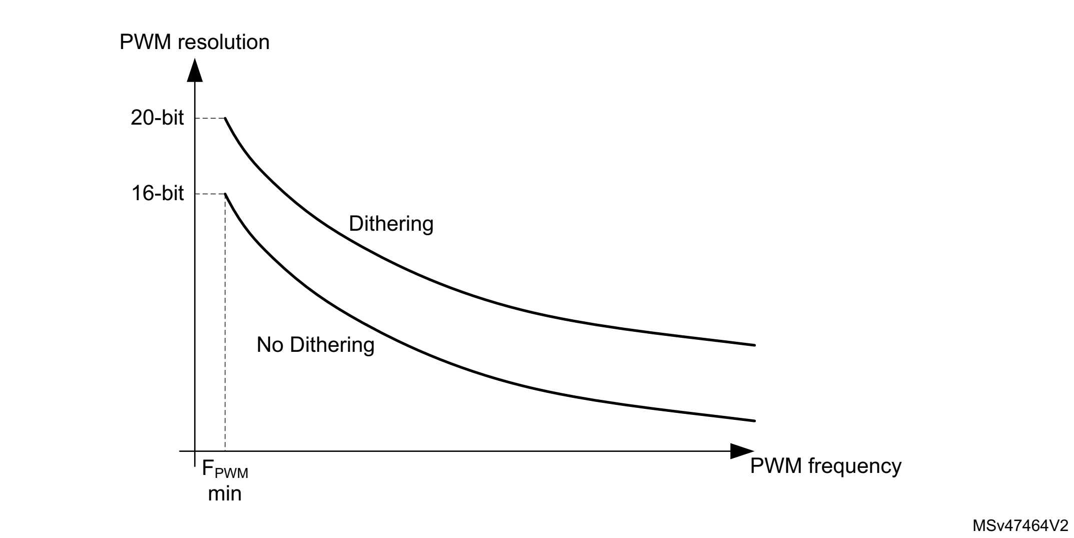
$$\text{Dithering mode disabled: } F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65536}$$

$$\text{Dithering mode enabled: } F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

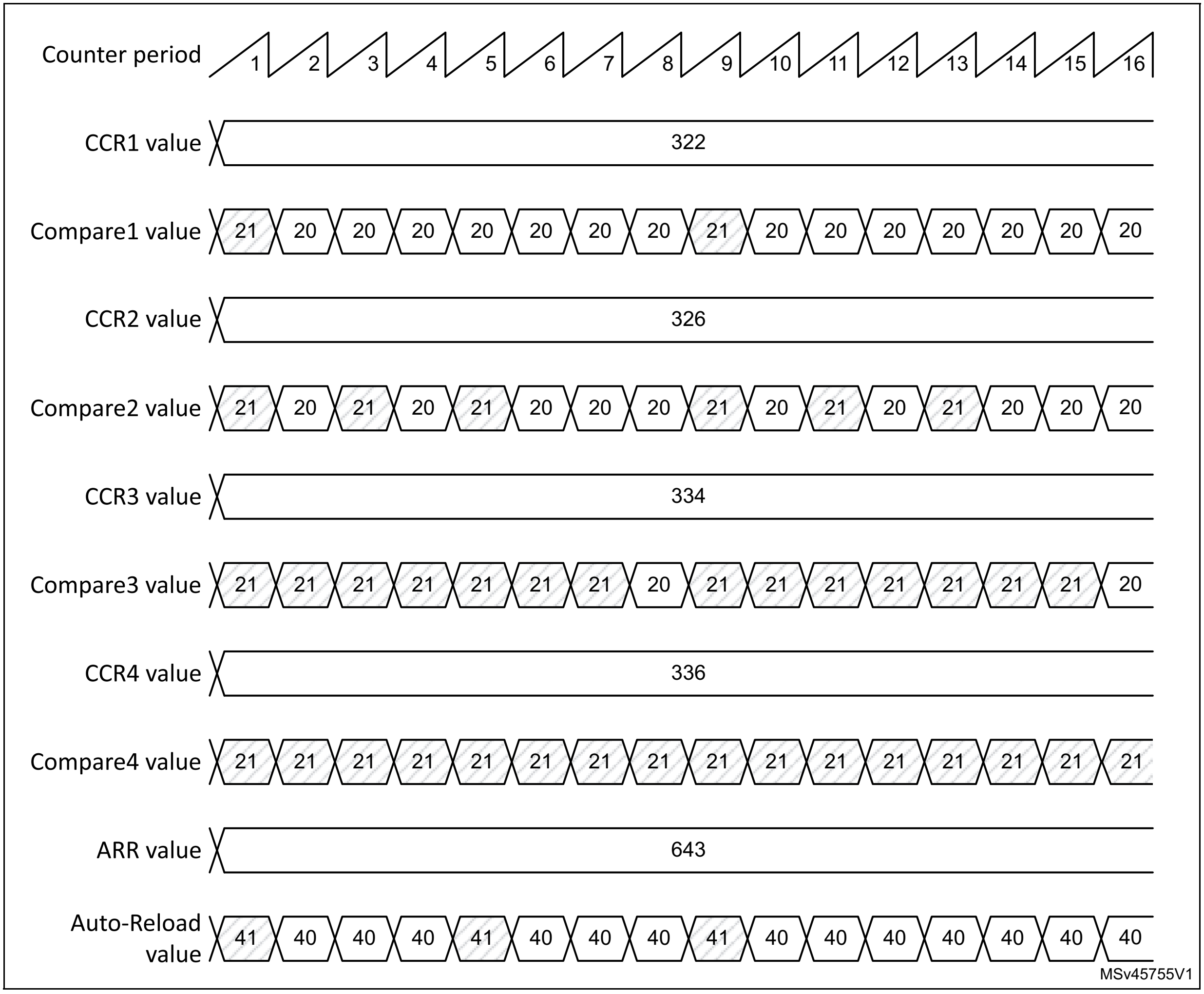
As shown on the *Figure 306* below, the dithering mode allows to increase the PWM resolution whatever the PWM frequency.

Figure 306. PWM resolution vs frequency



The duty cycle and / or period changes are spread over 16 consecutive periods, as described in the *Figure 307* below.

Figure 307. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 256* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 256. CCR and ARR register change dithering pattern

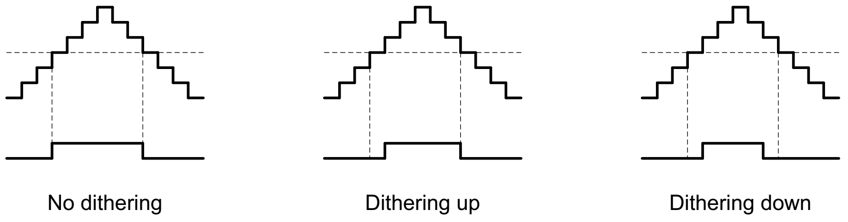
LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 256. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to '00'). In this case, the dithering pattern is applied over 8 consecutive PWM periods, considering the up and down counting phases as shown in the *Figure 308* below.

Figure 308. Dithering effect on duty cycle in center-aligned PWM mode



MSv50904V1

Table 257 below shows how the dithering pattern is added in center-aligned PWM mode.

Table 257. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 257. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

28.3.14 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:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

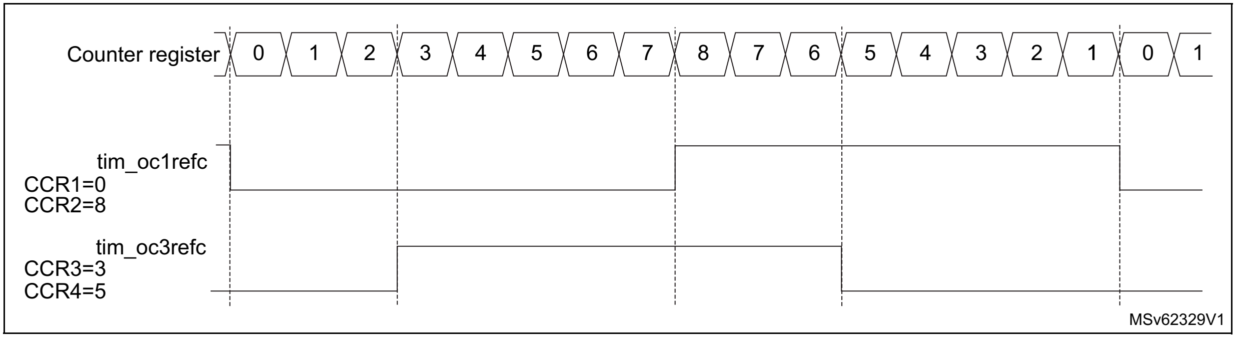
Asymmetric PWM mode can be selected independently on two channel (one tim_ocx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

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 tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 1.

Figure 309 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 309. Generation of 2 phase-shifted PWM signals with 50% duty cycle



28.3.15 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, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_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.

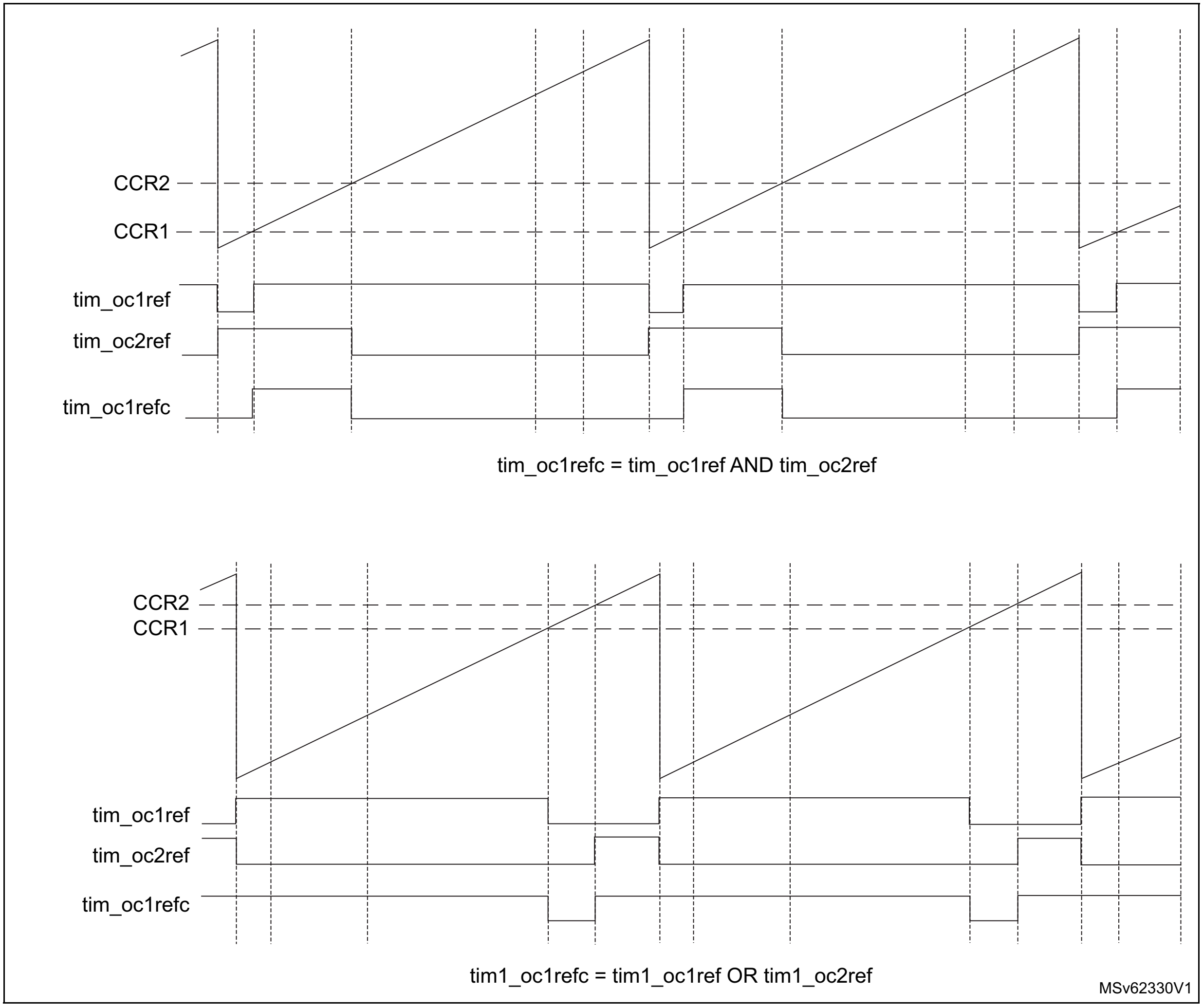
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 310 represents an example of signals that can be generated using combined 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 310. Combined PWM mode on channel 1 and 3



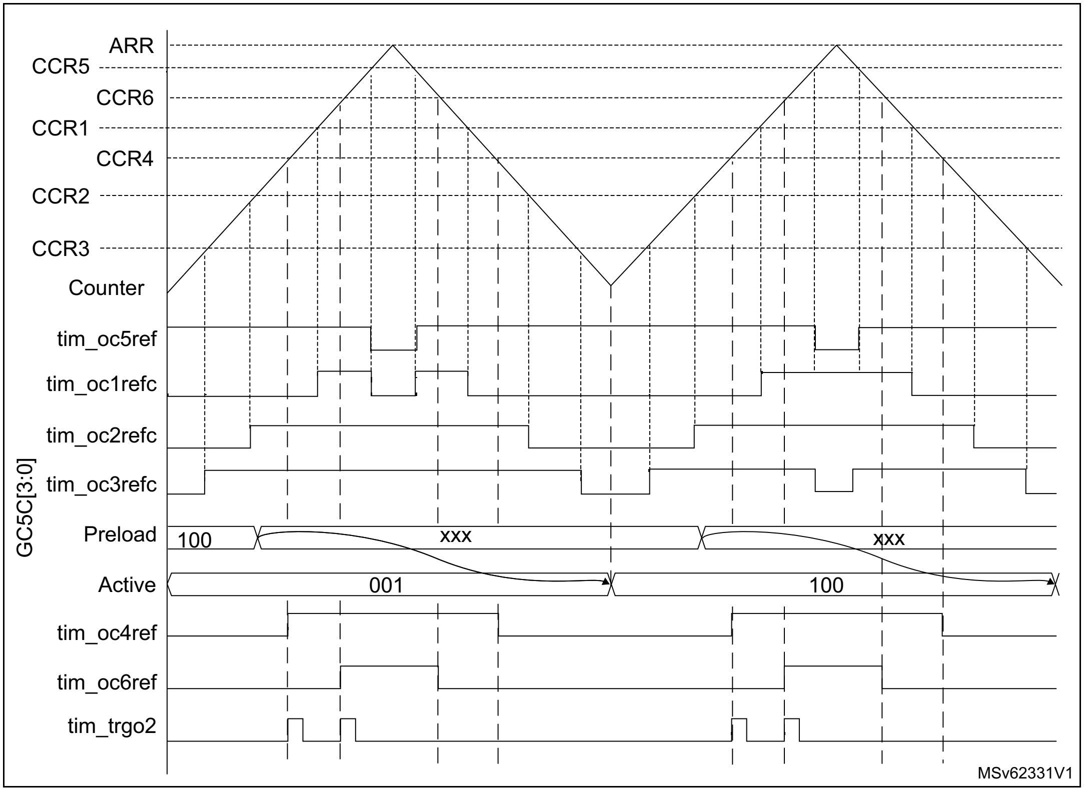
28.3.16 Combined 3-phase PWM mode

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 tim_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 tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, tim_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 311. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Please refer to *Section 28.3.31: ADC synchronization* for more details.

28.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8/TIM20) 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 tim_ocx or complementary tim_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 tim_ocx and tim_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 265: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature on page 1203* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

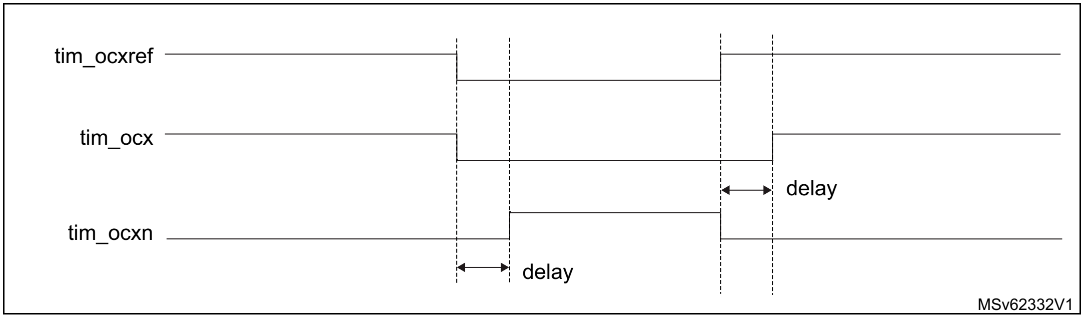
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates 2 outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_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 tim_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 (tim_ocx or tim_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 tim_ocxref. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 312. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIMx_DTR2 allows to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 313*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN=1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 313. Asymmetrical deadtime

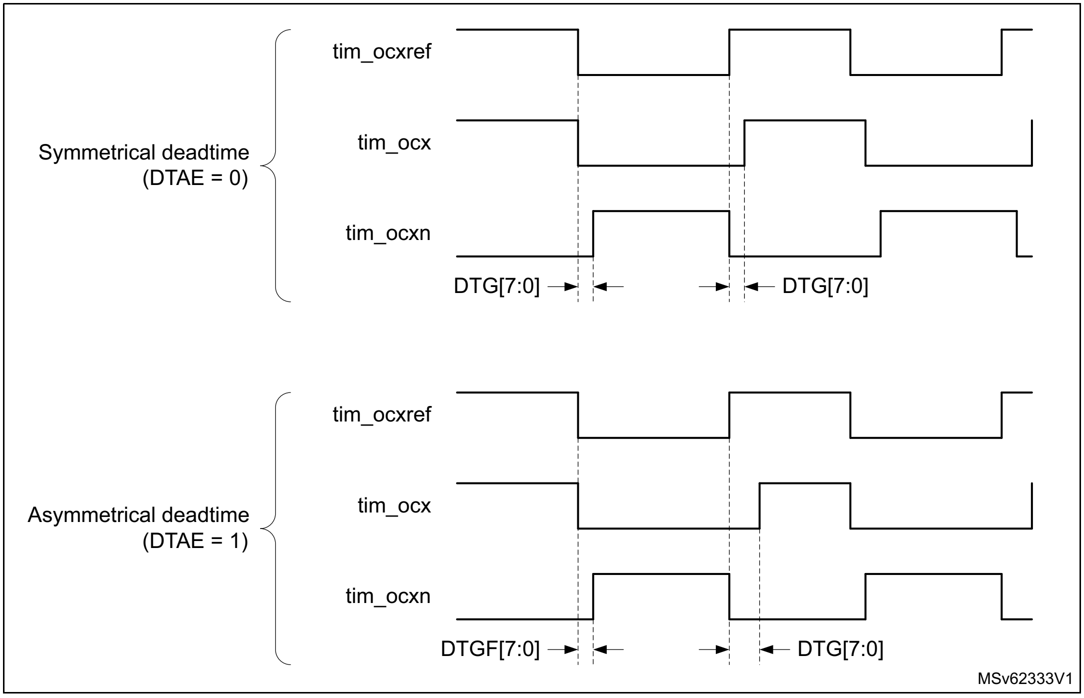


Figure 314. Dead-time waveforms with delay greater than the negative pulse

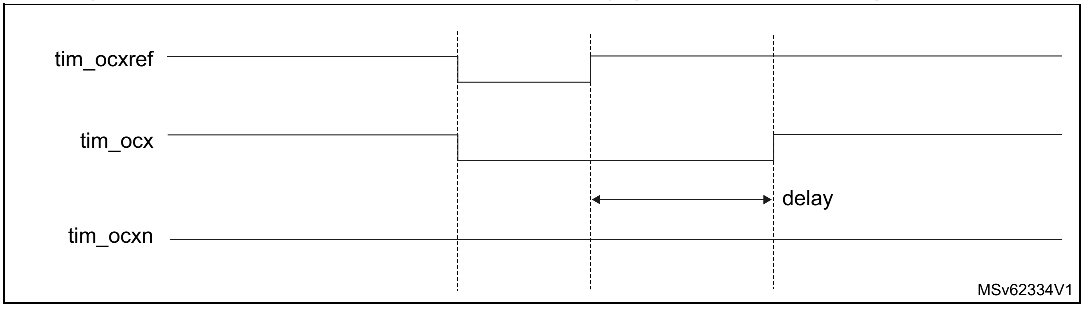
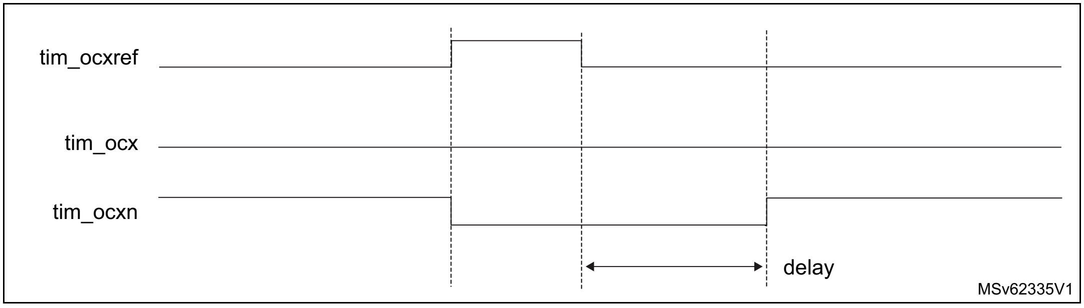


Figure 315. Dead-time waveforms with delay greater than the positive pulse



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 28.6.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8, 20)* for delay calculation.

Re-directing tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be re-directed to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows to send a specific waveform (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 tim_ocxn is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP=0 then tim_ocxn=tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE=CCxNE=1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

28.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. 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, ECC / parity errors,...) 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 to enable /disable the outputs 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 tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 265: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature on page 1203* 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. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx 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 sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation)

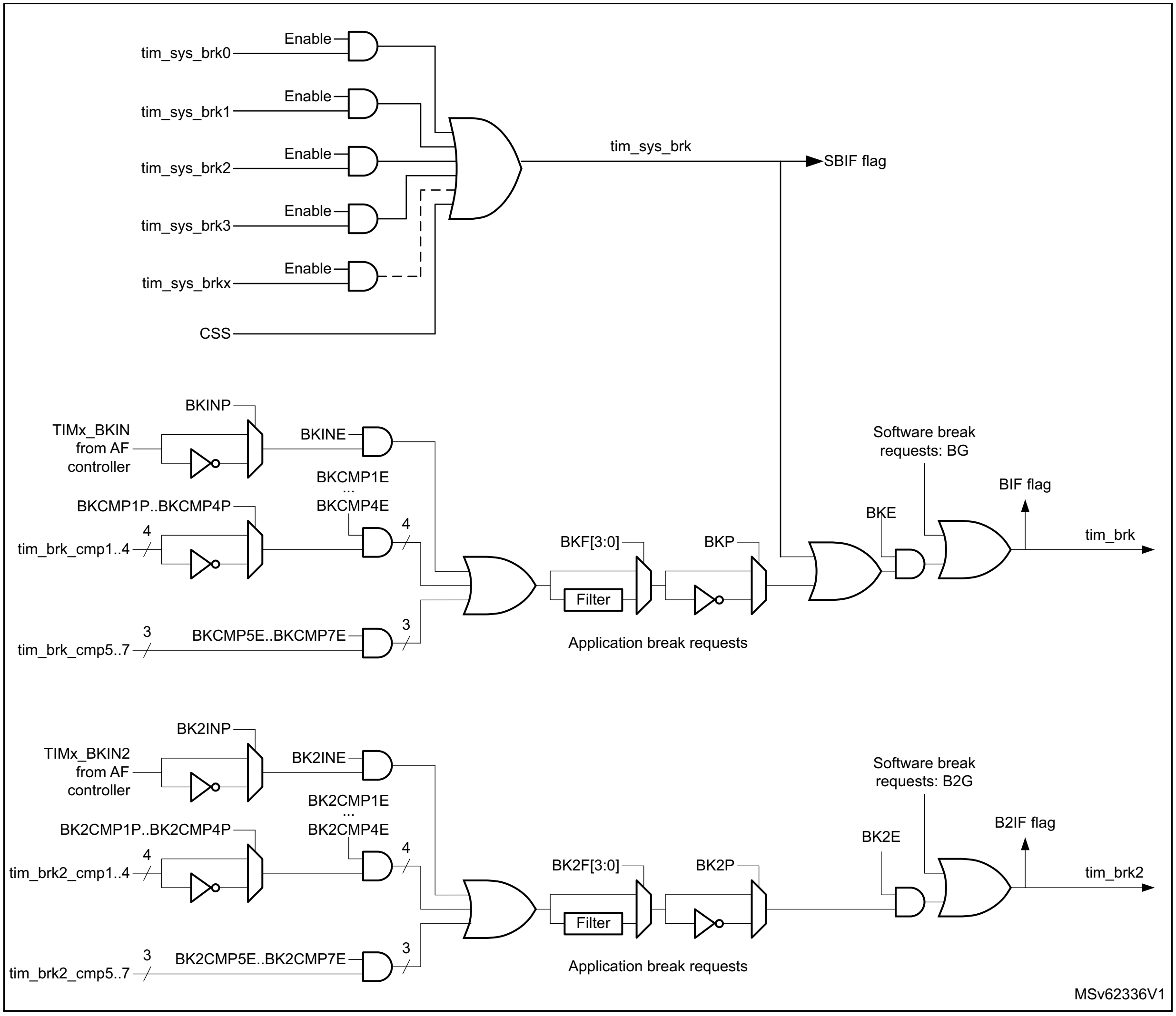
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 316* below.

Figure 316. 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, tim_ocx and tim_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 tim_ker_ck 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.

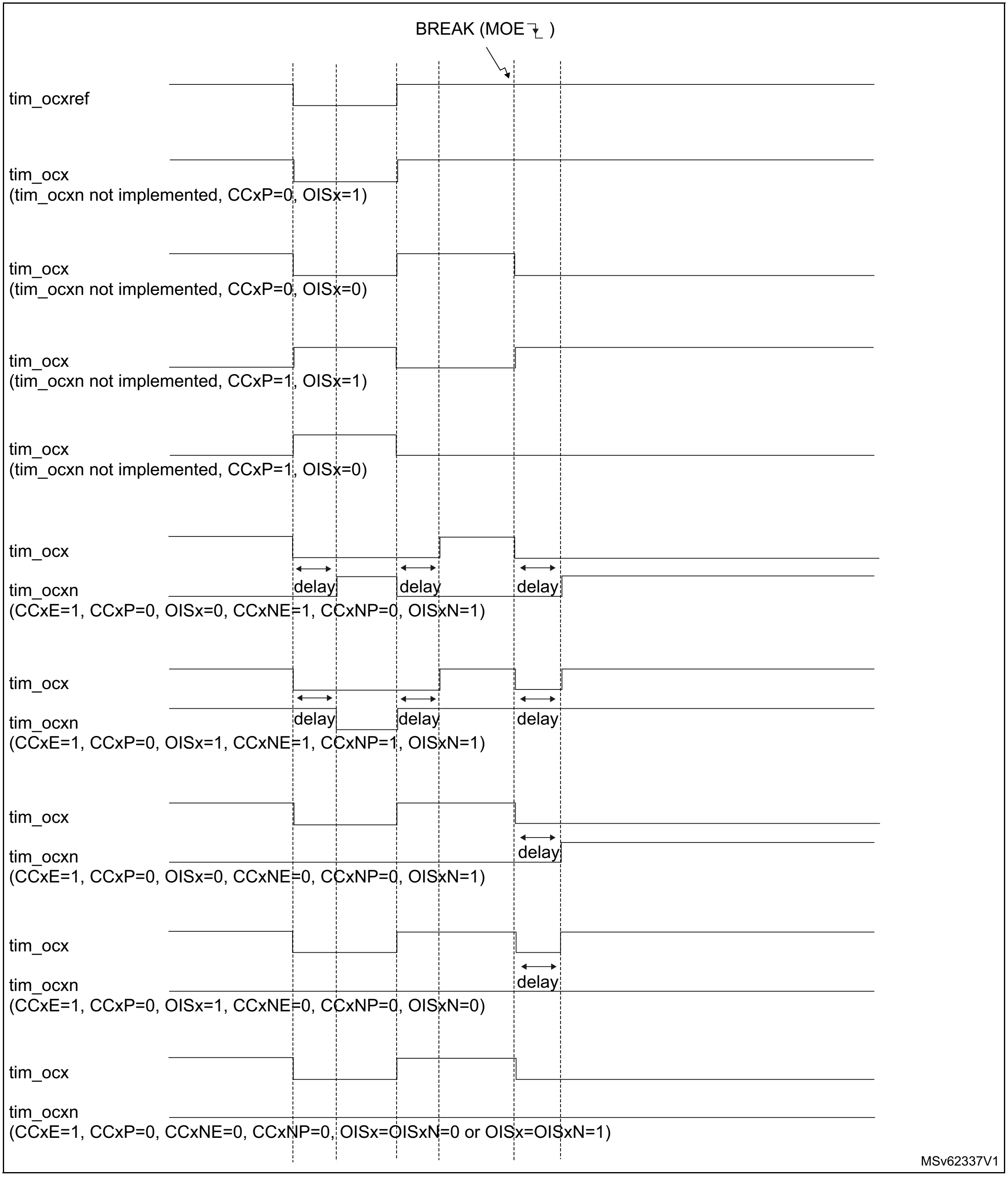
- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically 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: 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 to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_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 28.6.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8, 20)*. The LOCK bits can be written only once after an MCU reset.

Figure 317 shows an example of behavior of the outputs in response to a break.

Figure 317. Various output behavior in response to a break event on tim_brk (OSSI = 1)


BREAK (MOE ↓)
tim_ocxref
tim_ocx
(tim_ocxn not implemented, CCxP=0, OISx=1)
tim_ocx
(tim_ocxn not implemented, CCxP=0, OISx=0)
tim_ocx
(tim_ocxn not implemented, CCxP=1, OISx=1)
tim_ocx
(tim_ocxn not implemented, CCxP=1, OISx=0)
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, OISx=0, CCxNE=1, CCxNP=0, OISxN=1)
delay
delay
delay
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=1, OISxN=1)
delay
delay
delay
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISxN=1)
delay
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, OISx=1, CCxNE=0, CCxNP=0, OISxN=0)
delay
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, CCxNE=0, CCxNP=0, OISx=OISxN=0 or OISx=OISxN=1)
MSv62337V1

The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 258*.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 258. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_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 318 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 318. PWM output state following tim_brk and tim_brk2 assertion (OSSI=1)

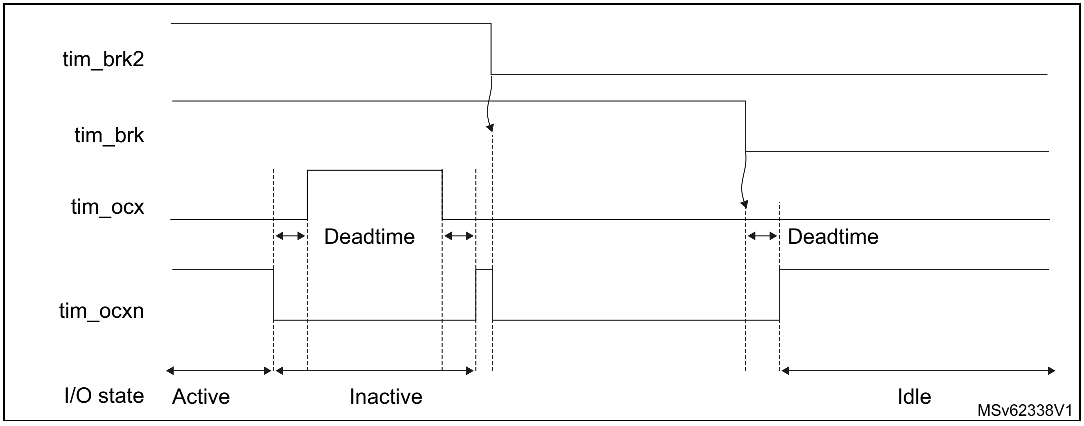
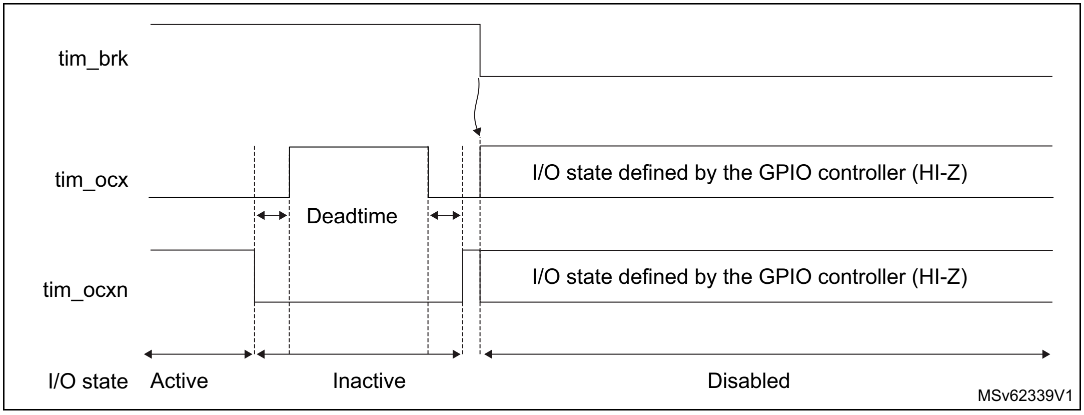


Figure 319. PWM output state following tim_brk assertion (OSSI=0)


tim_brk
tim_ocx
tim_ocxn
Deadtime
I/O state defined by the GPIO controller (HI-Z)
I/O state defined by the GPIO controller (HI-Z)
I/O state
Active
Inactive
Disabled
MSv62339V1

28.3.19 Bidirectional break inputs

The TIM1/TIM8/TIM20 are featuring bidirectional break I/Os, as represented on *Figure 320*.

This provides support for:

- 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 sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The tim_brk and tim_brk2 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 tim_brk and tim_brk2 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 is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKIN and TIMx_BKIN2 I/Os.

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 BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 259*).

Table 259. Break protection disarming conditions

MOE	BKBID (BK2BID)	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

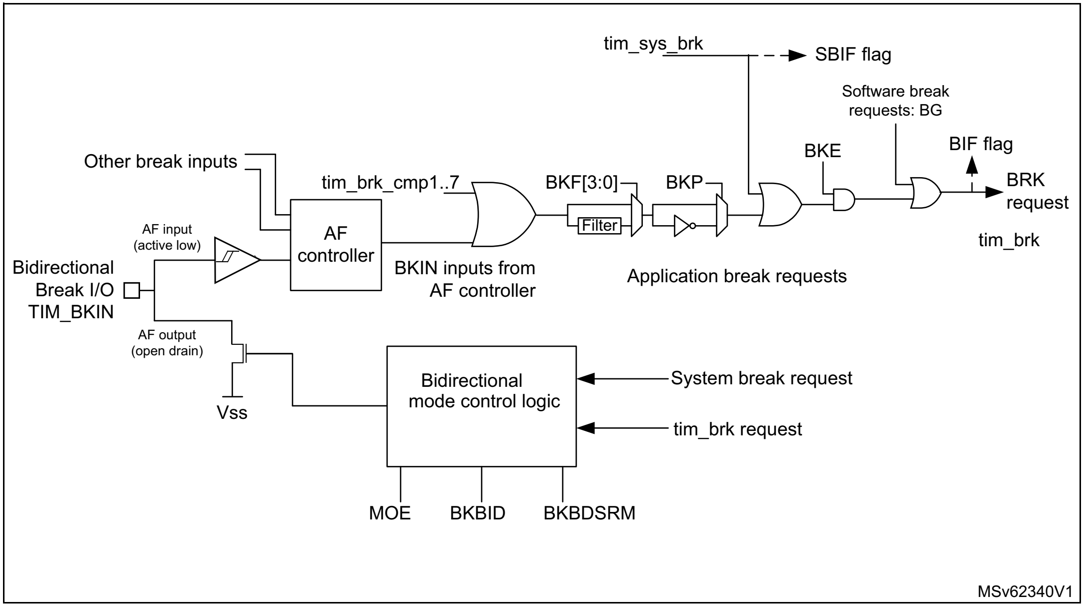
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

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 320. Output redirection (tim_brk2 request not represented)

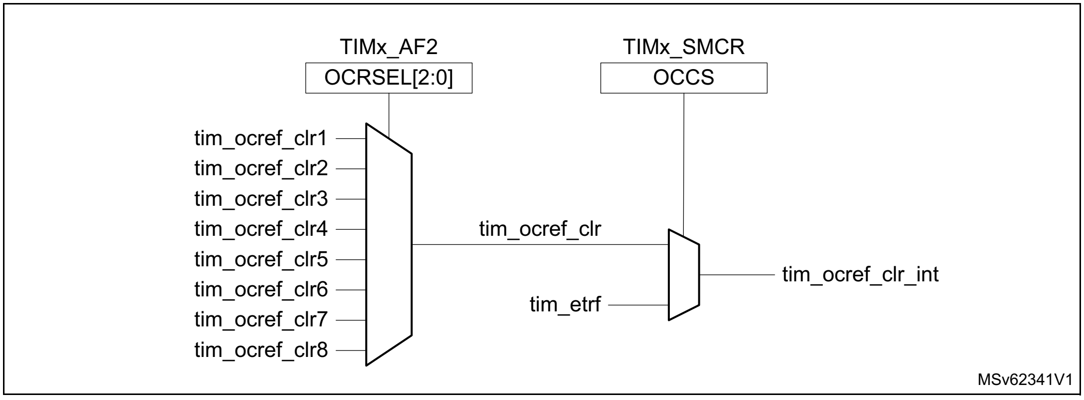


28.3.20 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_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. tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

The tim_ocref_clr input can be selected among several inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown on the *Figure 321* below. Refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for a list of sources available in the product.

Figure 321. tim_ocref_clr input selection multiplexer

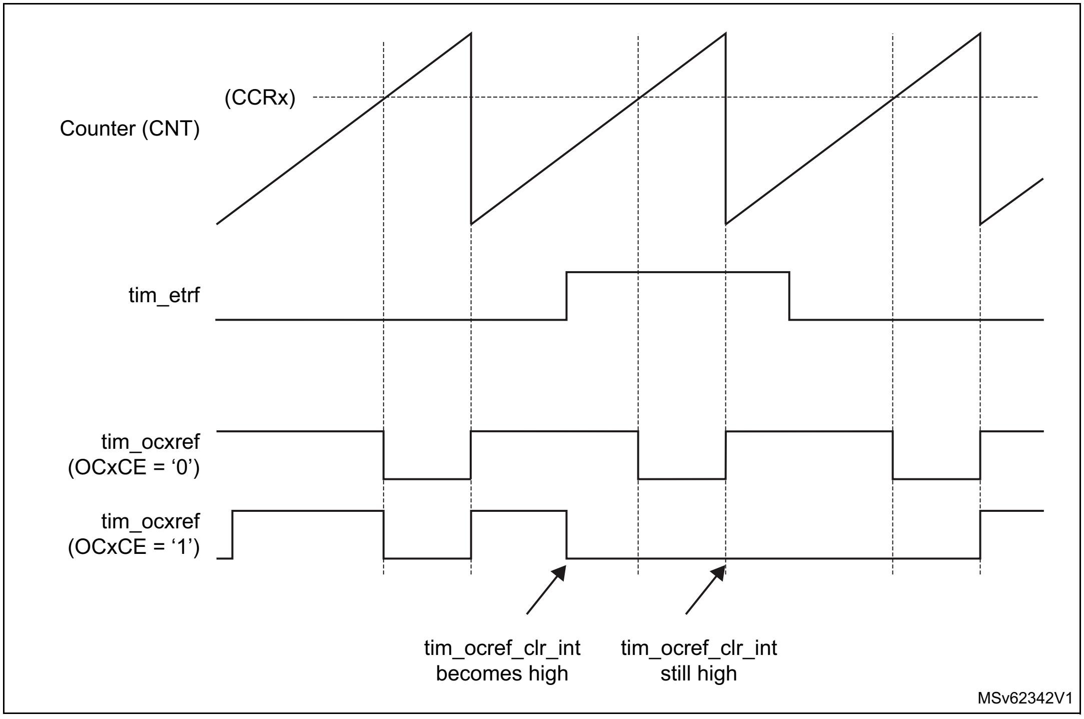


When tim_etrf is chosen, tim_etr_in must be configured as follows:

1. The External Trigger Prescaler must 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 application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 322 shows the behavior of the tim_ocxref signal when the tim_etrf Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 322. Clearing TIMx tim_ocxref



Note: *In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.*

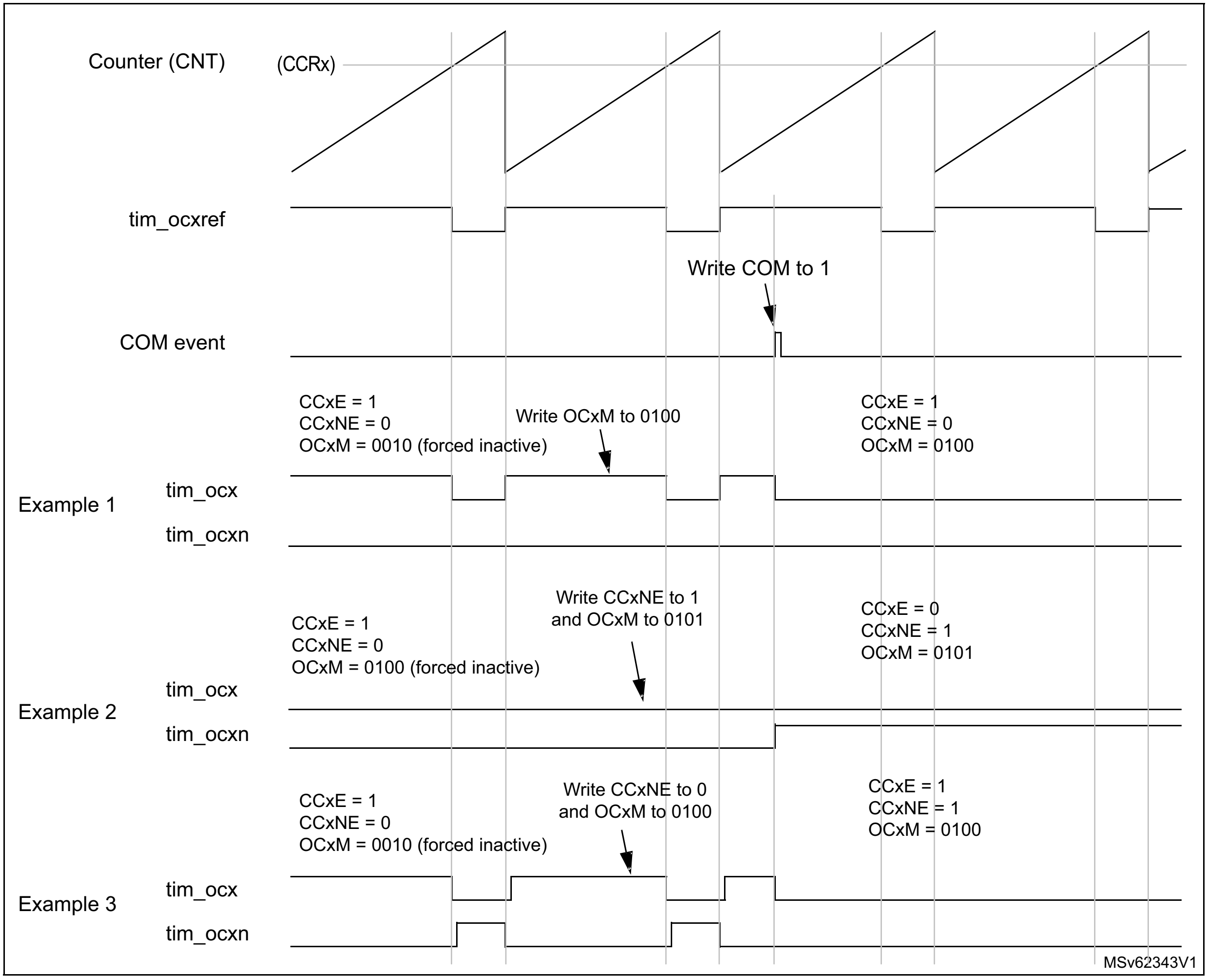
28.3.21 6-step PWM generation

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 323* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 323. 6-step generation, COM example (OSSR=1)



28.3.22 One-pulse mode

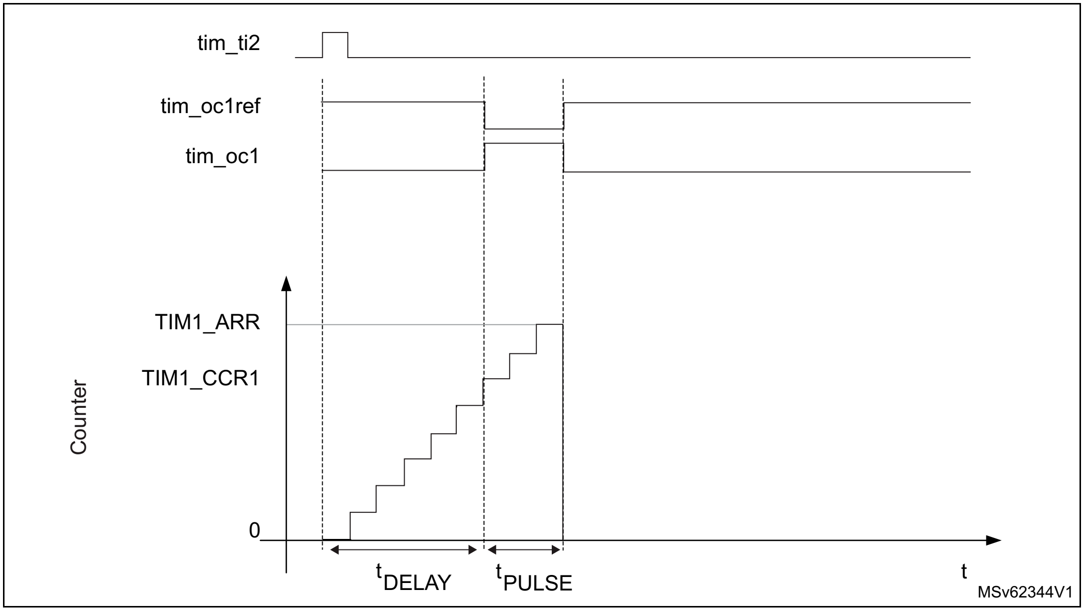
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 324. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S='01' in the TIMx_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS=00110 in the TIMx_SMCR register.
- tim_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 tim_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: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_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 tim_ocxref (and tim_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.

28.3.23 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 28.3.22*:

- 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

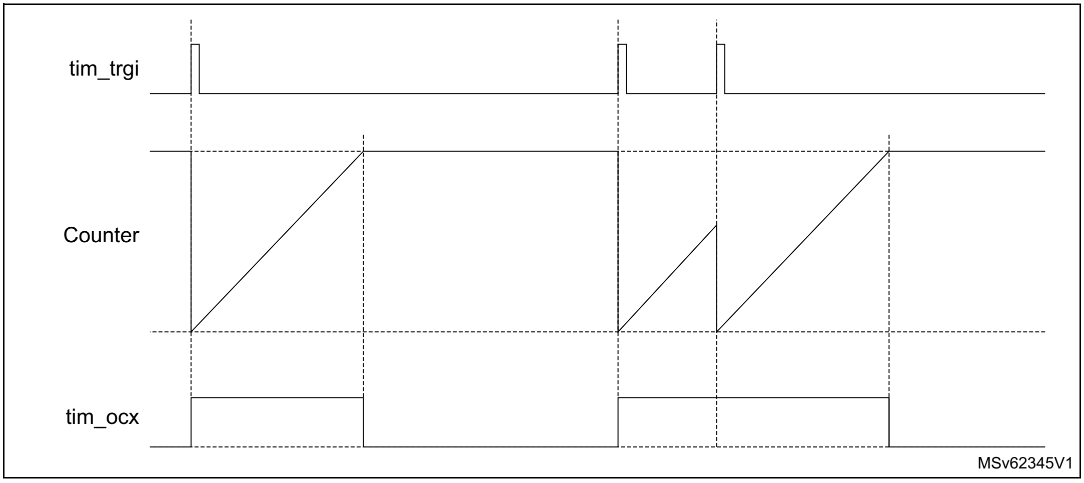
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for retriggerable OPM mode 1 or 2.

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: 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 325. Retriggerable one pulse mode

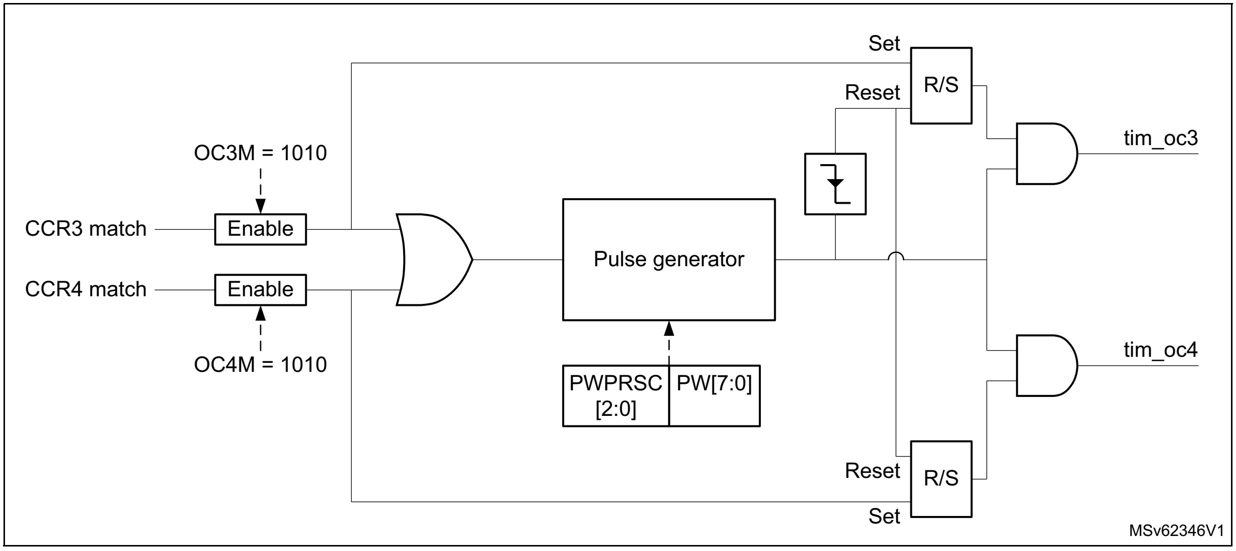


28.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

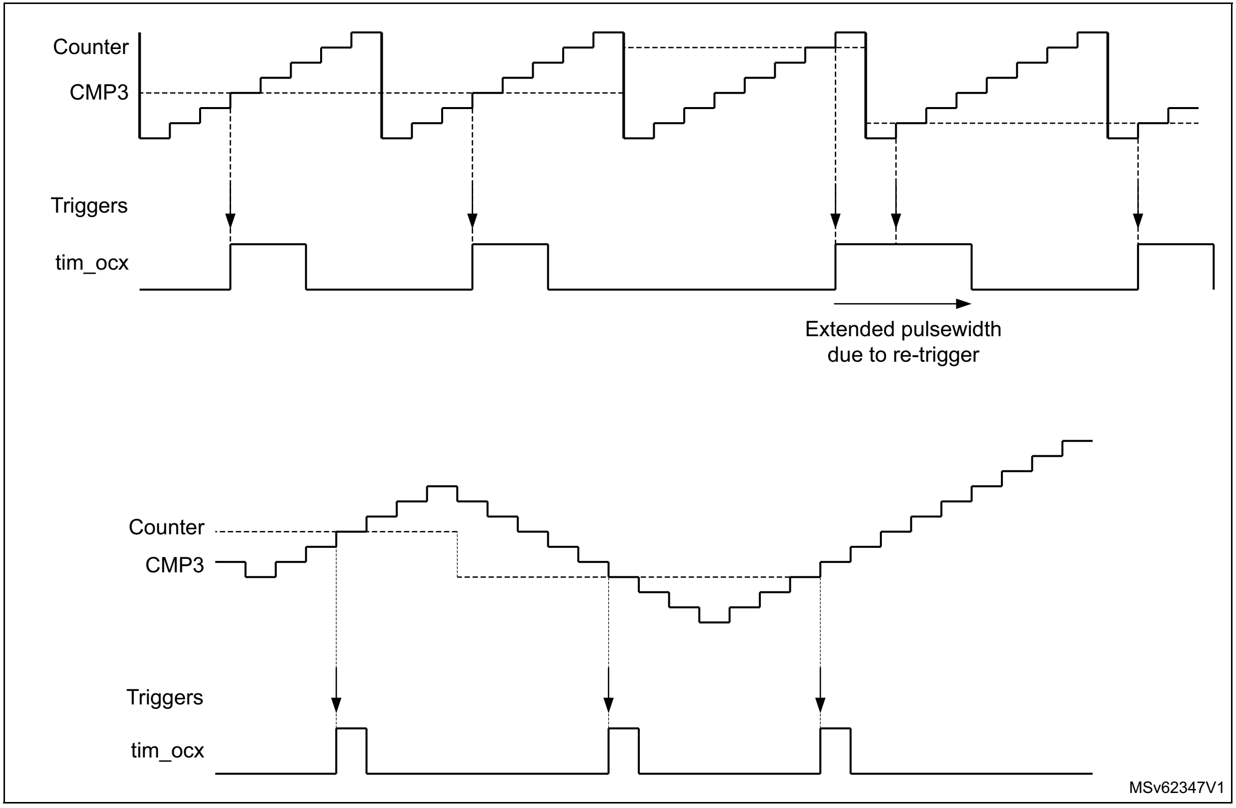
This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on the *Figure 326* below.

Figure 326. Pulse generator circuitry



The *Figure 327* below shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 327. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bit fields in TIMx_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

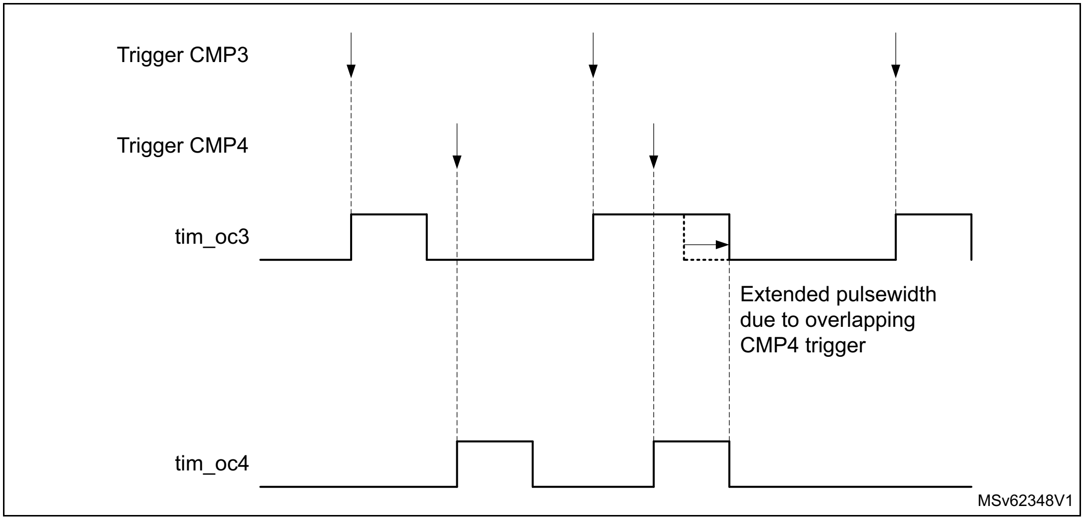
where t_{PWG} = $(2^{(PWPRSC[2:0])})$ x $t_{tim\_ker\_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is on-going, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the 1st arriving trigger is extended (because of the re-trigger), while the pulse width of the last arriving trigger is correct (as shown on the Figure 328 below).

Figure 328. Extended pulsewidth in case of concurrent triggers



28.3.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS=‘0001’ in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS=’0010’ if it is counting on tim_ti2 edges only and SMS=’0011’ if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_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 tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 260*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1=tim_ti1 if not filtered and not inverted, tim_ti2fp2=tim_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 (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_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, 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.

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 tim_ti1 and tim_ti2 do not switch at the same time.

Table 260. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	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 the external trigger input and trigger a counter reset.

The *Figure 329* 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, tim_ti1fp1 mapped on tim_ti1).
- CC2S='01' (TIMx_CCMR2 register, tim_ti1fp2 mapped on tim_ti2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, tim_ti1fp1 non-inverted, tim_ti1fp1=tim_ti1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, tim_ti1fp2 non-inverted, tim_ti1fp2= tim_ti2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 329. Example of counter operation in encoder interface mode.

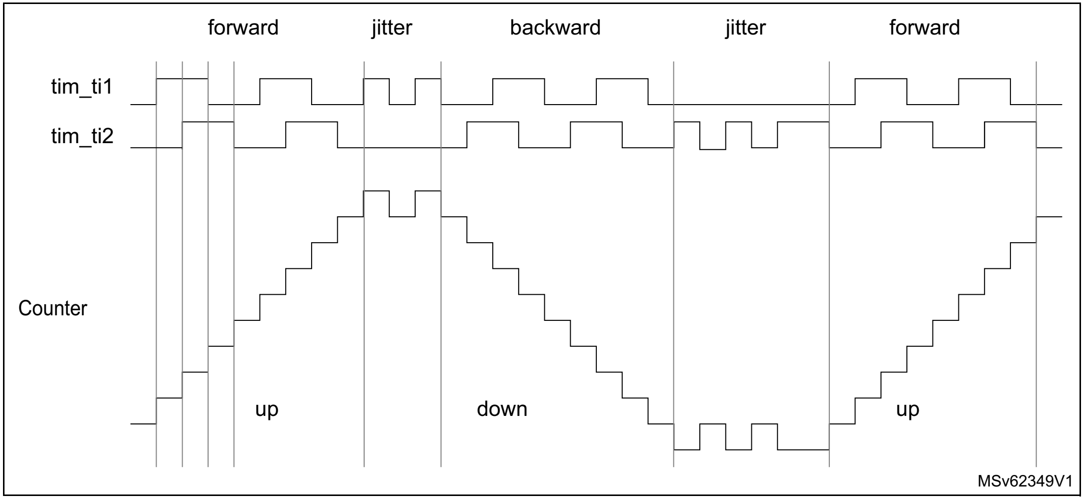
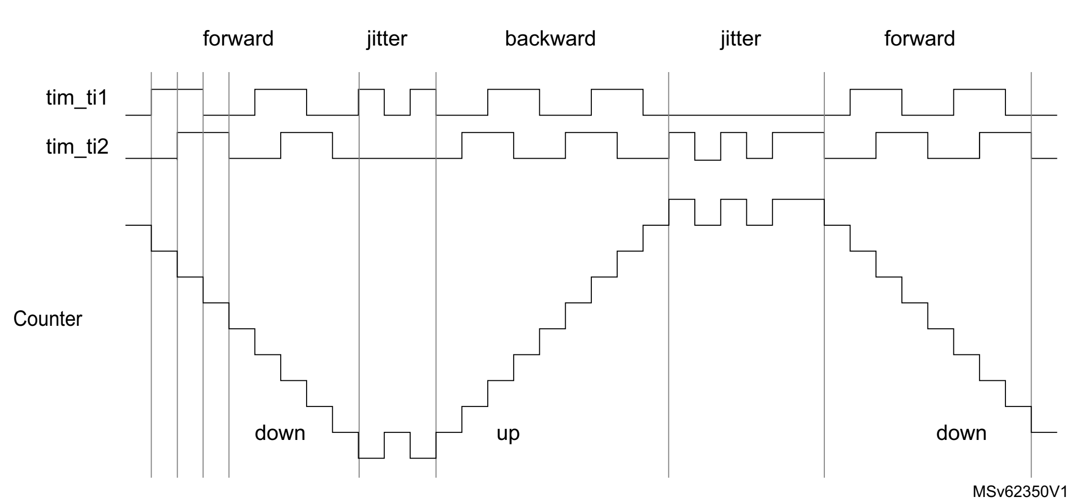


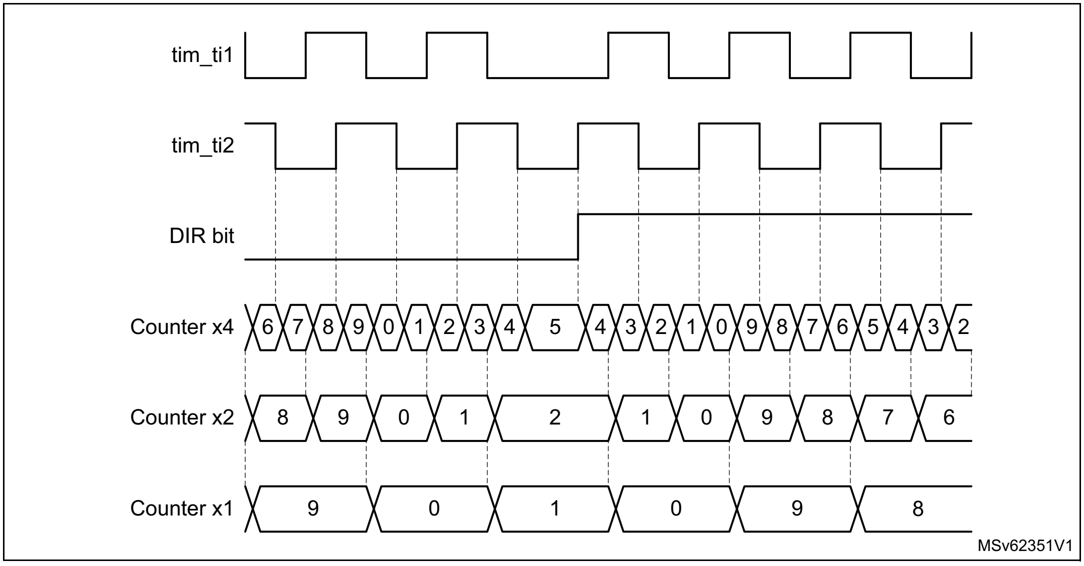
Figure 330 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P='1').

Figure 330. Example of encoder interface mode with tim_ti1fp1 polarity inverted.



The *Figure 331* below shows the timer counter value during a speed reversal, for various counting modes.

Figure 331. Quadrature encoder counting modes



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).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support other types of encoders.

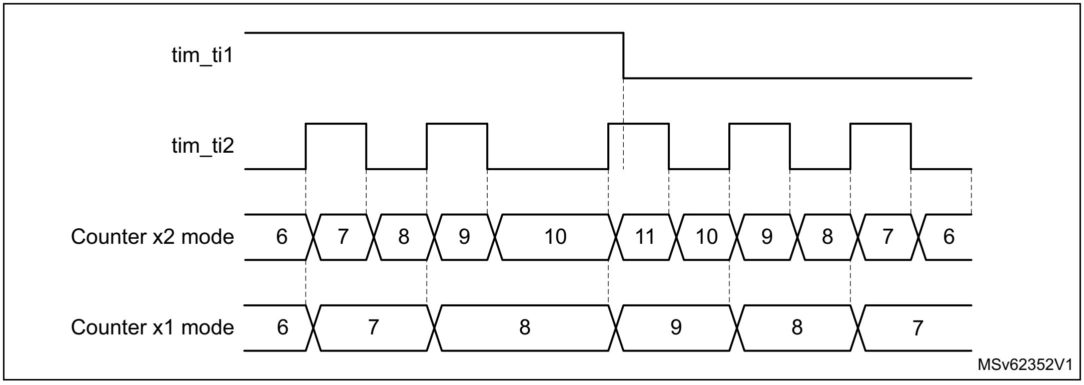
In the "clock plus direction" mode shown on *Figure 332*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 332. Direction plus clock encoder mode



Directional Clock encoder mode

In the “directional clock” mode on *Figure 333*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 333* below) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 334* below).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 333* below), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 334* below).

Figure 333. Directional clock encoder mode (CC1P = CC2P = 0)

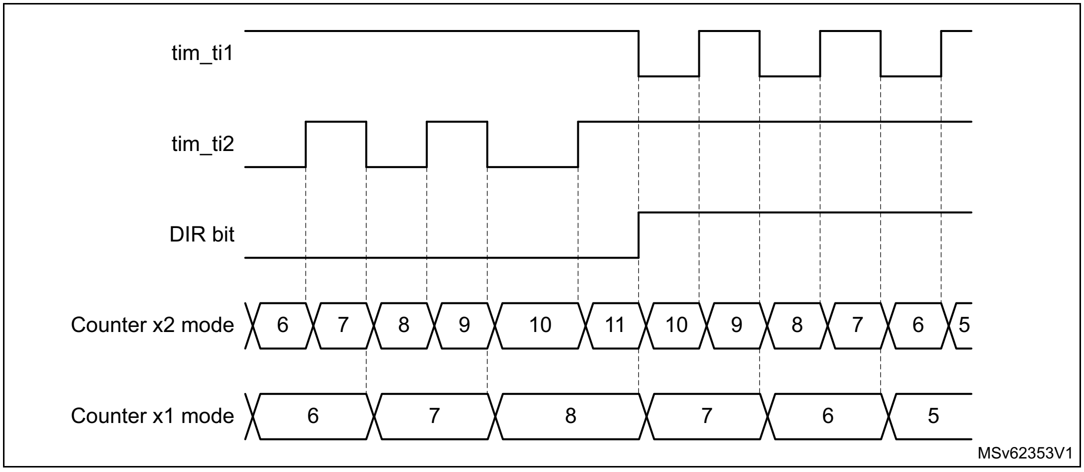
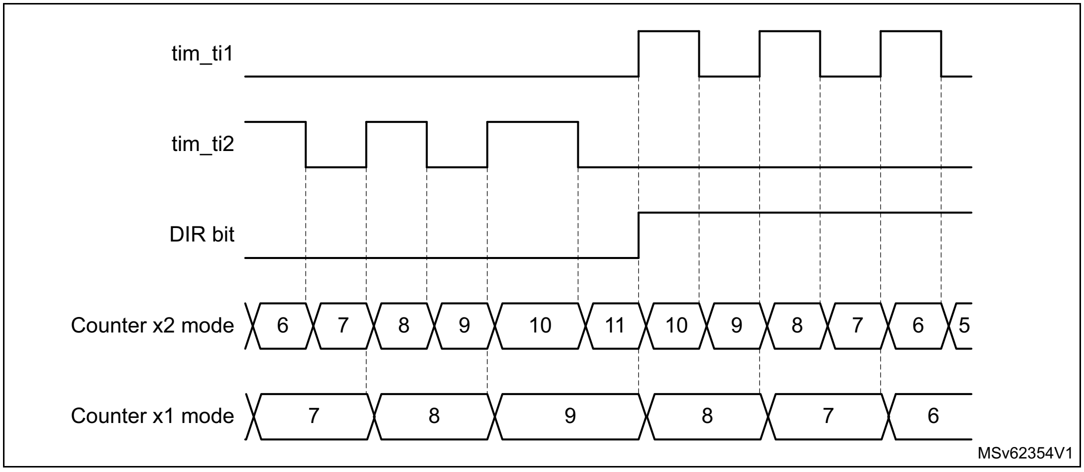


Figure 334. Directional clock encoder mode (CC1P = CC2P = 1)



The *Table 261* here-below details how the directional clock mode operates, for any input transition.

Table 261. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP=0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP=1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP=0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP=1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index Input

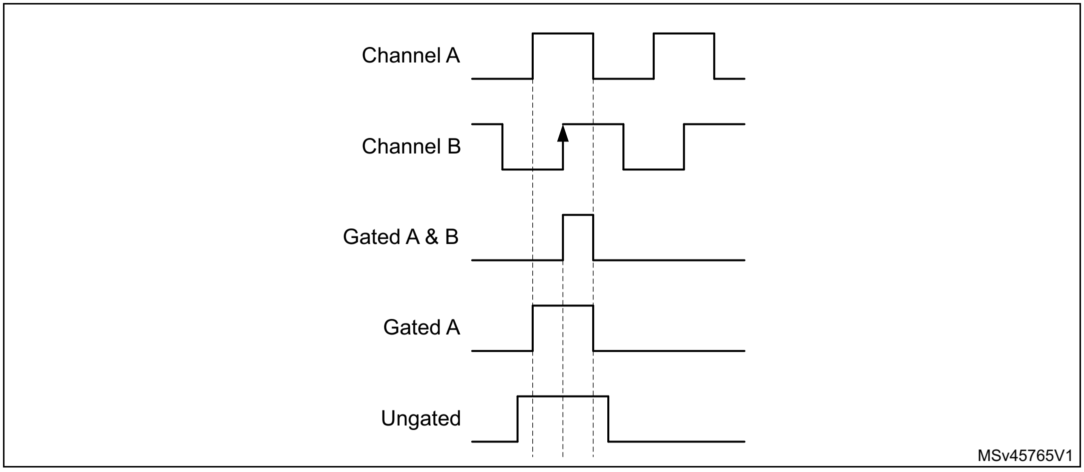
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The Index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMX_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Commercially available encoders are proposed with several options for index pulse conditioning, as per the *Figure 335* below:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

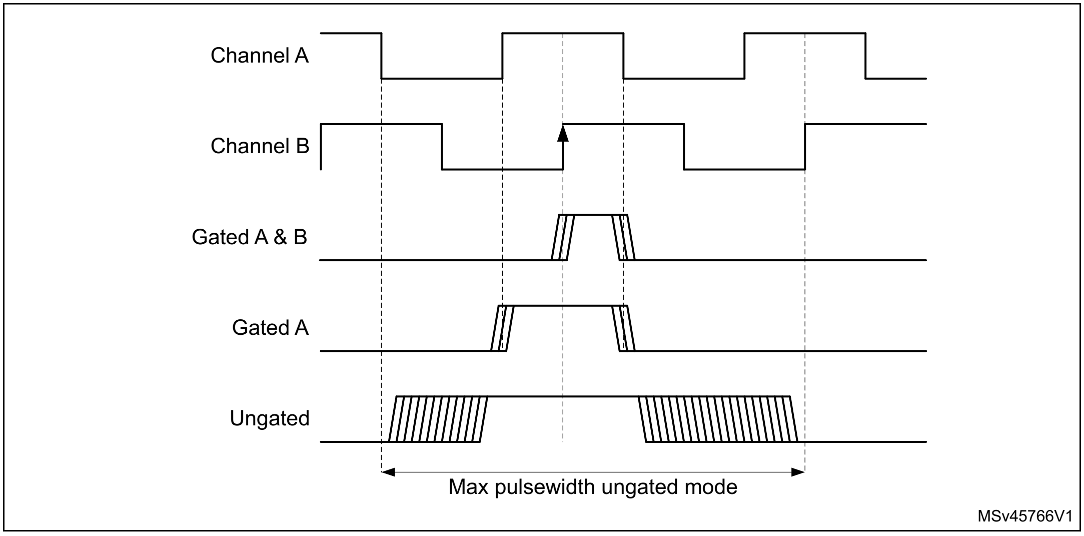
Figure 335. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 336* below.

In ungated mode, the signal must be strictly below 2 encoder periods. If the pulsewidth is greater or equal to 2 encoder period, the counter is reset multiple times.

Figure 336. Jittered Index signals



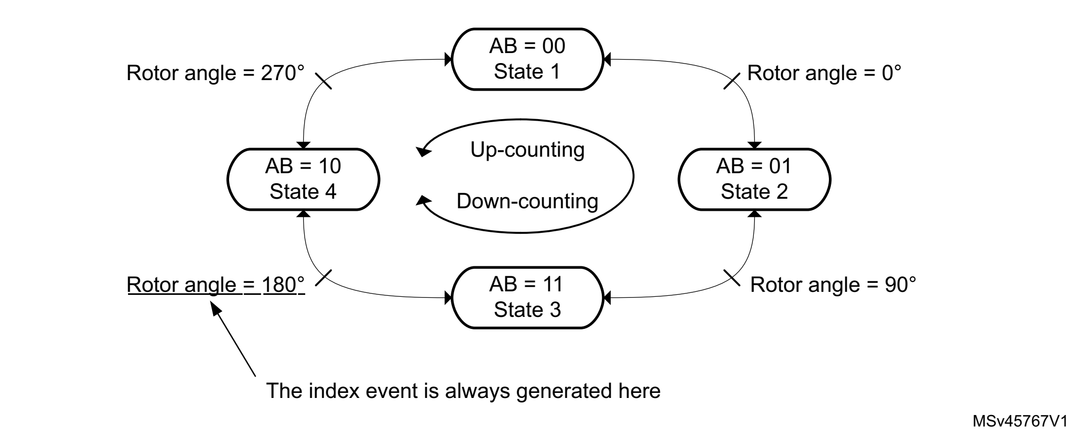
The timer supports the 3 gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (i.e. channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The Index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. The *Figure 337* below shows at which position is the index generated, for a simplistic example (an encoder providing 4 edges par mechanical rotation).

Figure 337. Index generation for IPOS[1:0] = 11



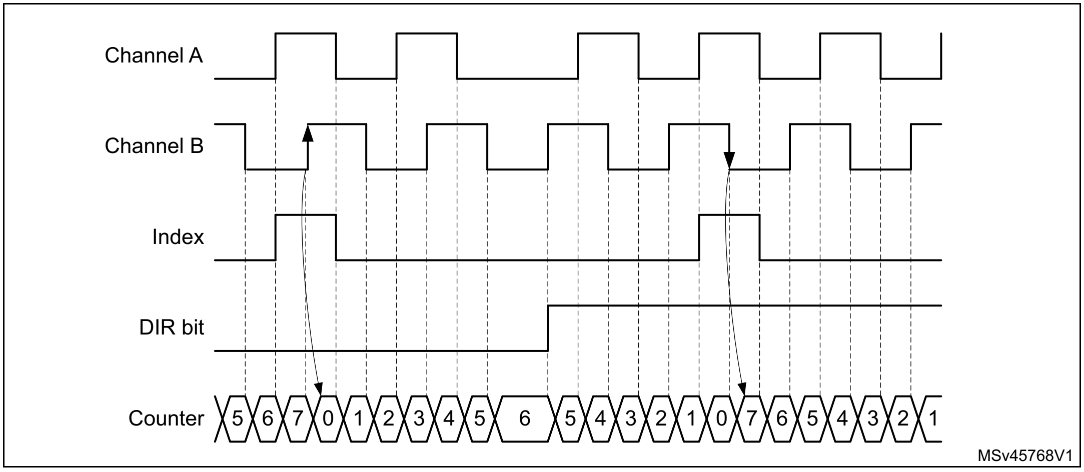
The *Figure 338* below presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is '11' (ChA=1, ChB=1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the '11' state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

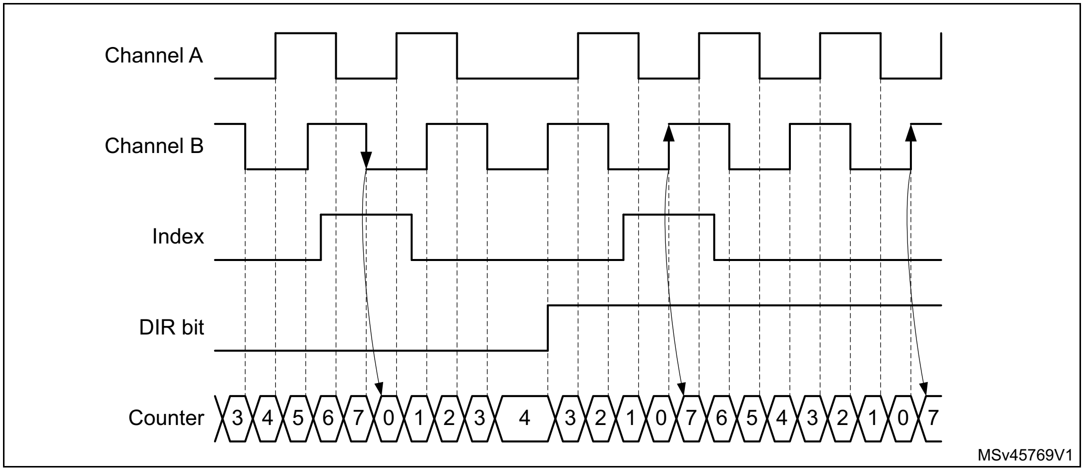
The arrows are indicating on which transition is the index event interrupt generated.

Figure 338. Counter reading with index gated on channel A (IPOS[1:0] = 11)



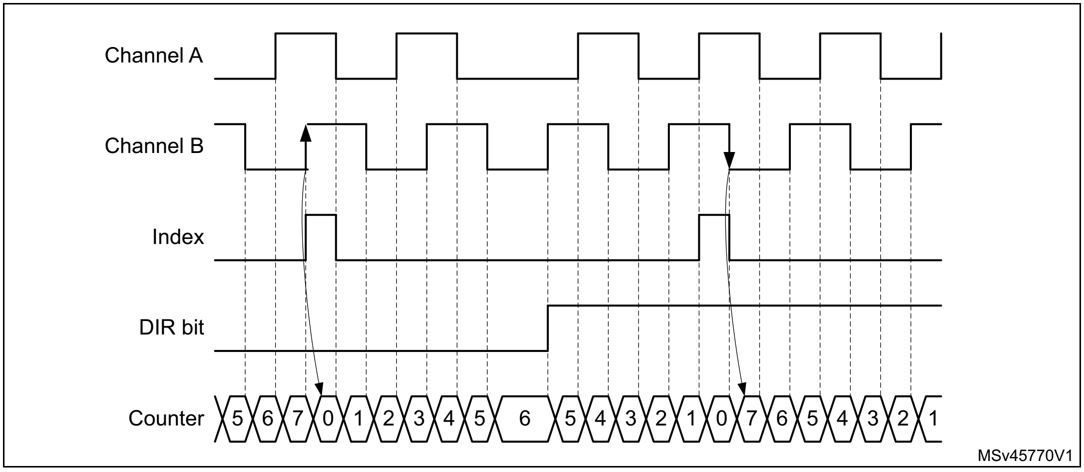
The *Figure 339.* below presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 339. Counter reading with index ungated (IPOS[1:0] = 00)



The *Figure 340.* below shows how the ‘gated on A & B’ mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 340. Counter reading with index gated on channel A and B



The *Figure 341* and *Figure 342* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 341. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

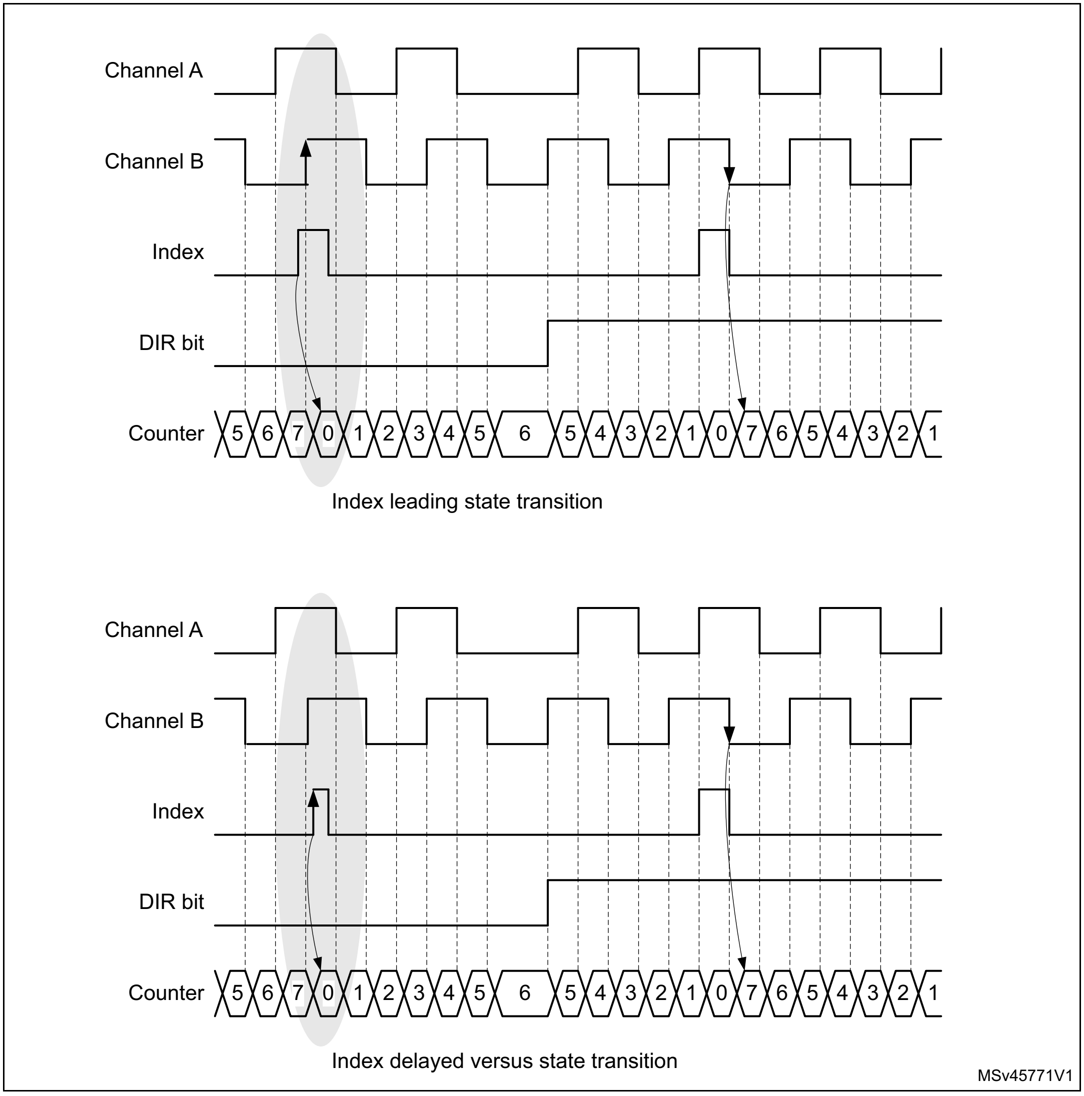
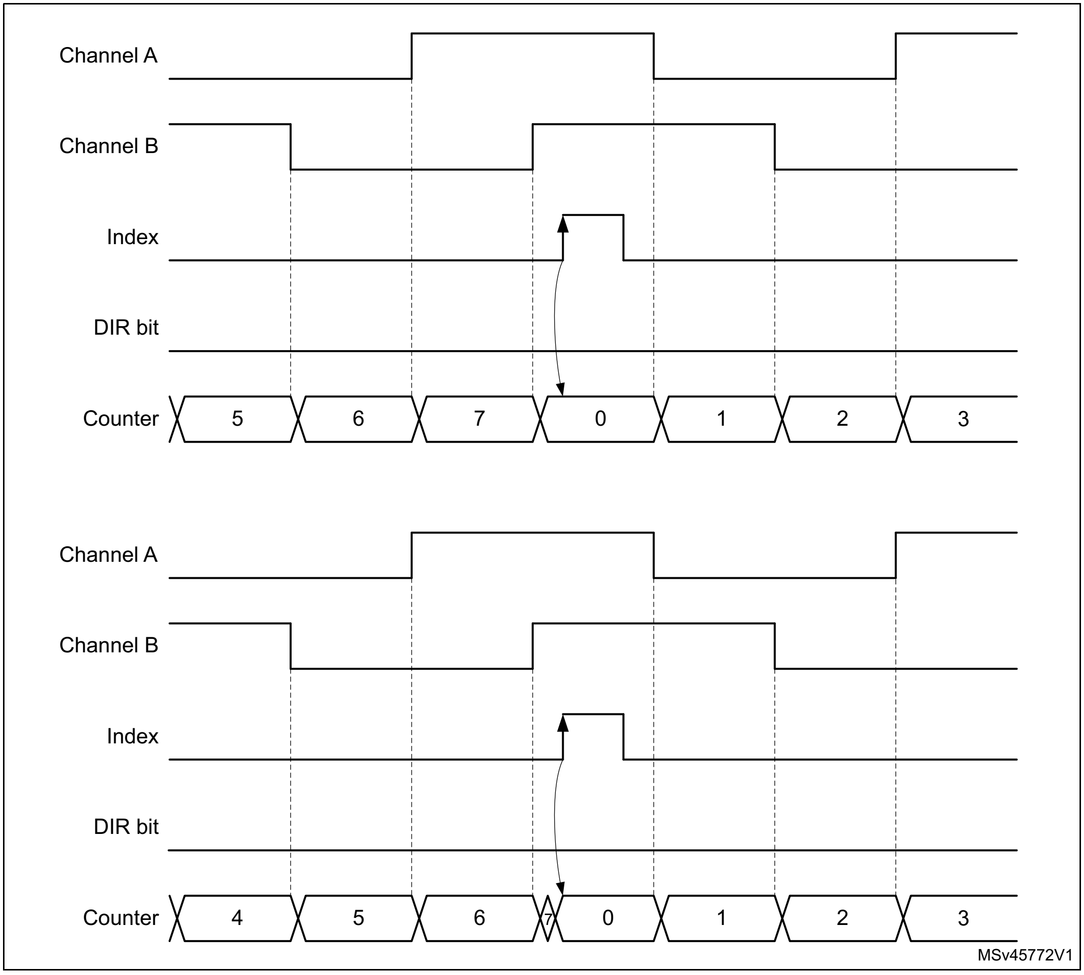
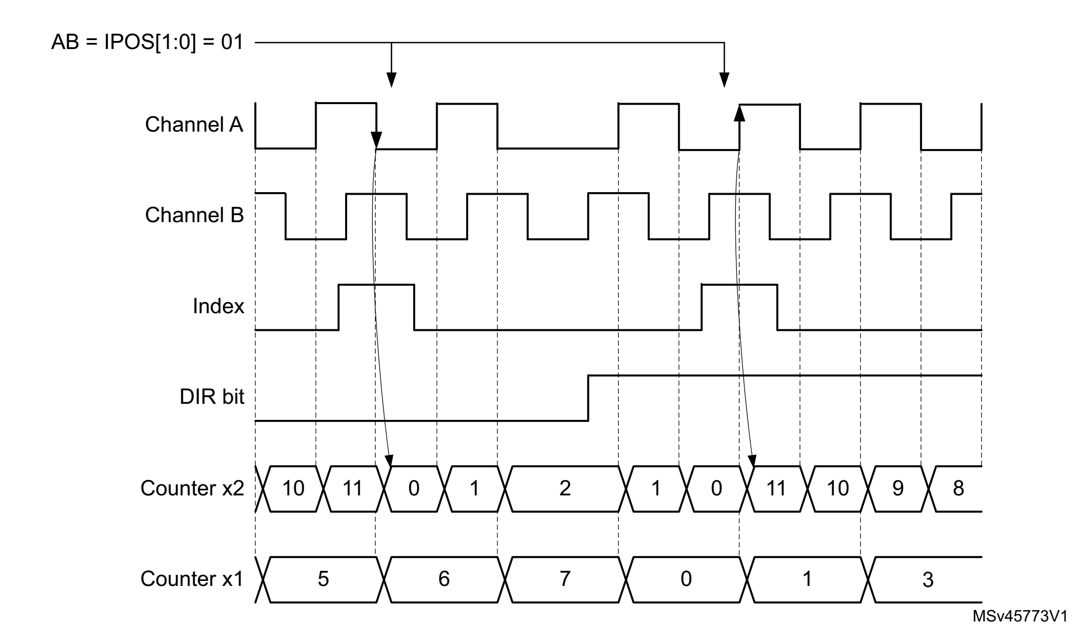


Figure 342. Counter reset Narrow index pulse (closer view, ARR = 0x07)



The *Figure 343* below shows how the index is managed in x1 and x2 modes.

Figure 343. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)



Directional index sensitivity

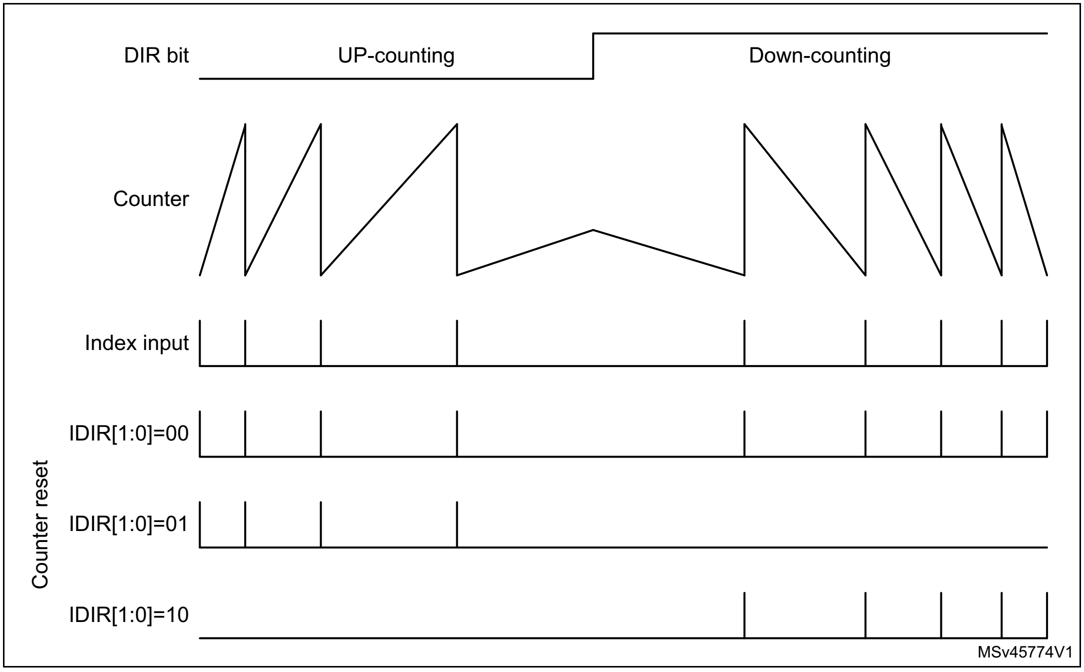
The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

The *Figure 344* below shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

Note: The IDR[1:0] bitfield must be written when IE bit is reset (index mode disabled).

Note: The directional index sensitivity is not supported in clock + direction mode. When SMS[3:0] = 1010 or 1011, the IDIR[1:0] must be set to 00.

Figure 344. Directional index sensitivity

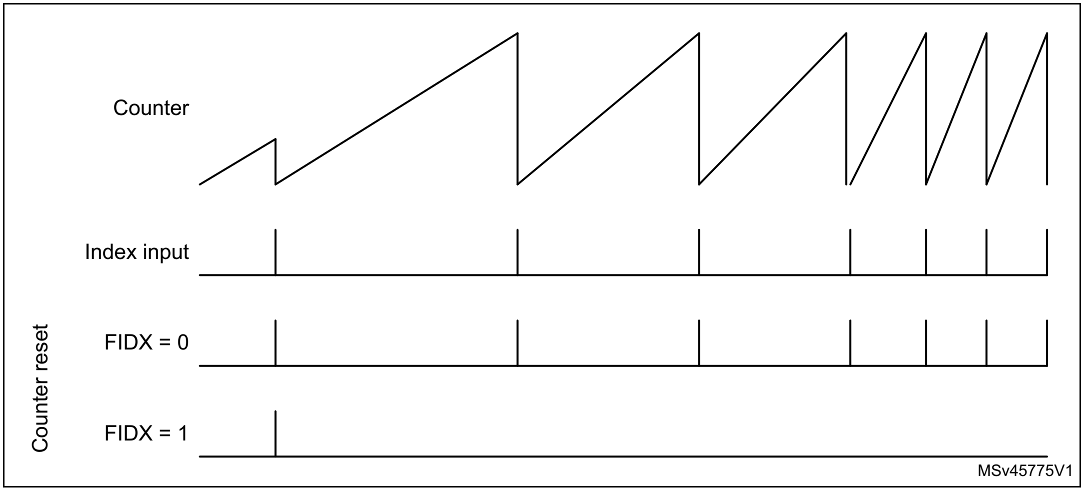


Special first index event management

The FIDX bit in the TIMx_ECR register allows the Index to be taken only once, as shown on the *Figure 345* below. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be re-armed by writing the FIDX bit to 0 and setting it again to 1.

Note: When FIDX = 1, the index can be issued twice (IDXF flag set) if the direction changes at position 0 (index active).

Figure 345. Counter reset as function of FIDX bit setting



Index management in non-quadrature mode

The *Figure 346* and *Figure 347* below detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 346. Index behavior in clock + direction mode, IPOS[0] = 1

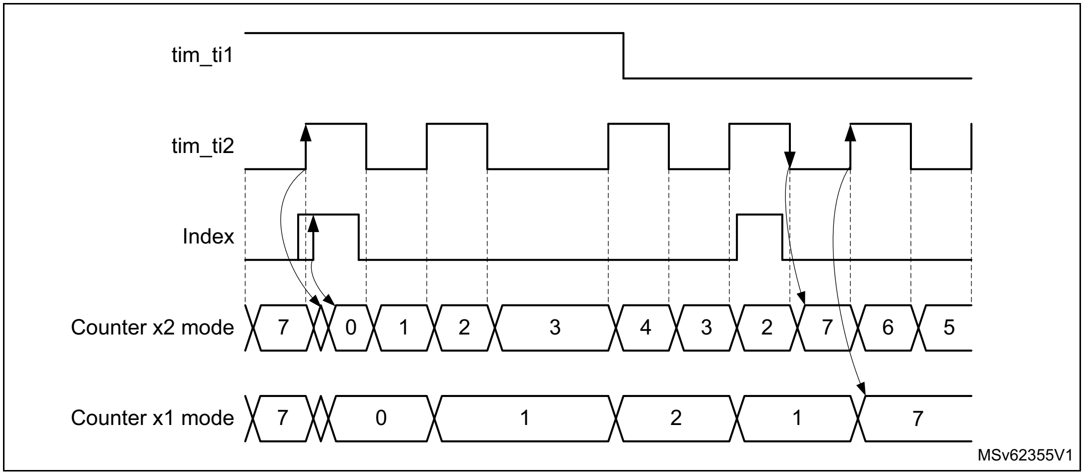
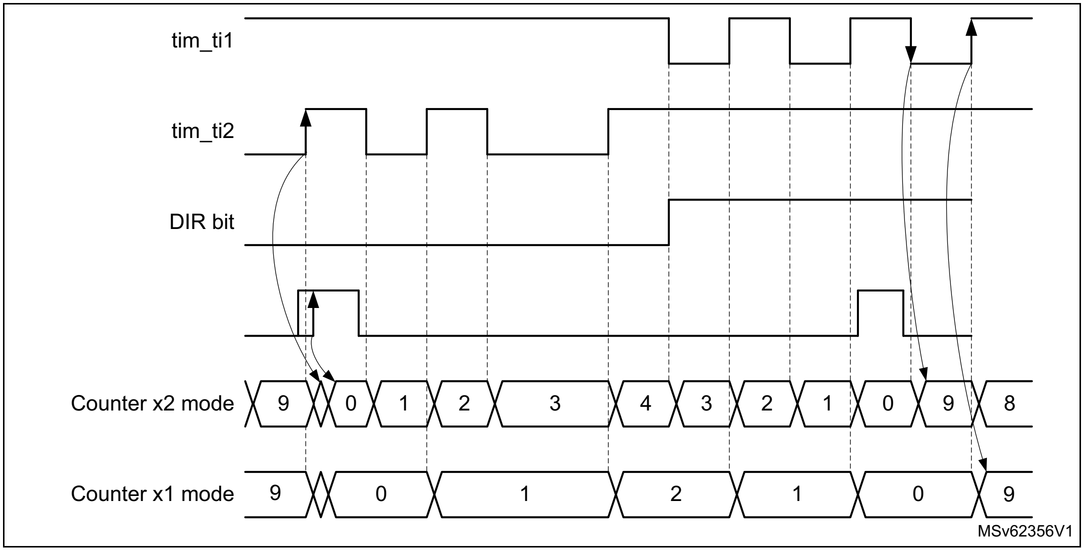


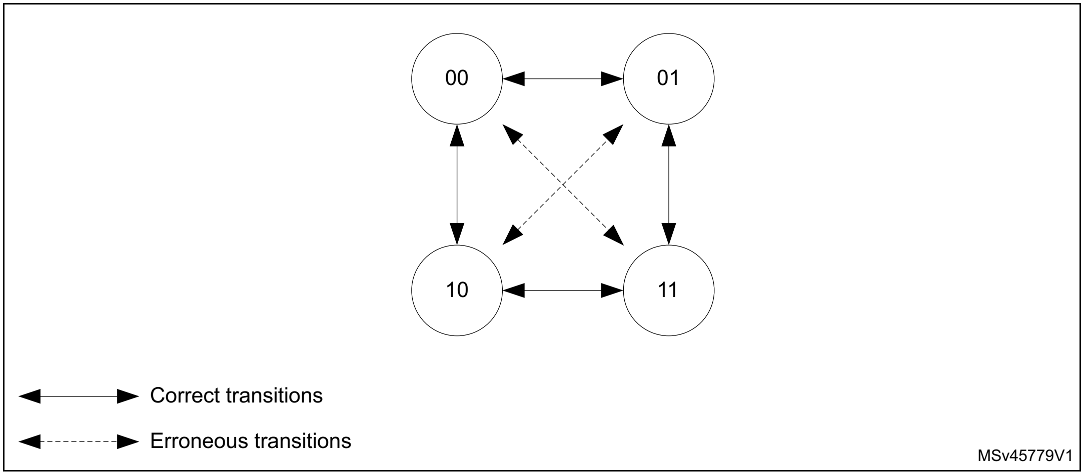
Figure 347. Index behavior in directional clock mode, IPOS[0] = 1



Encoder error management

For encoder configurations where 2 quadrature signals are available, it is possible to detect transition errors. The reading on the 2 inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on the *Figure 348.* below. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 348. State diagram for quadrature encoded signals



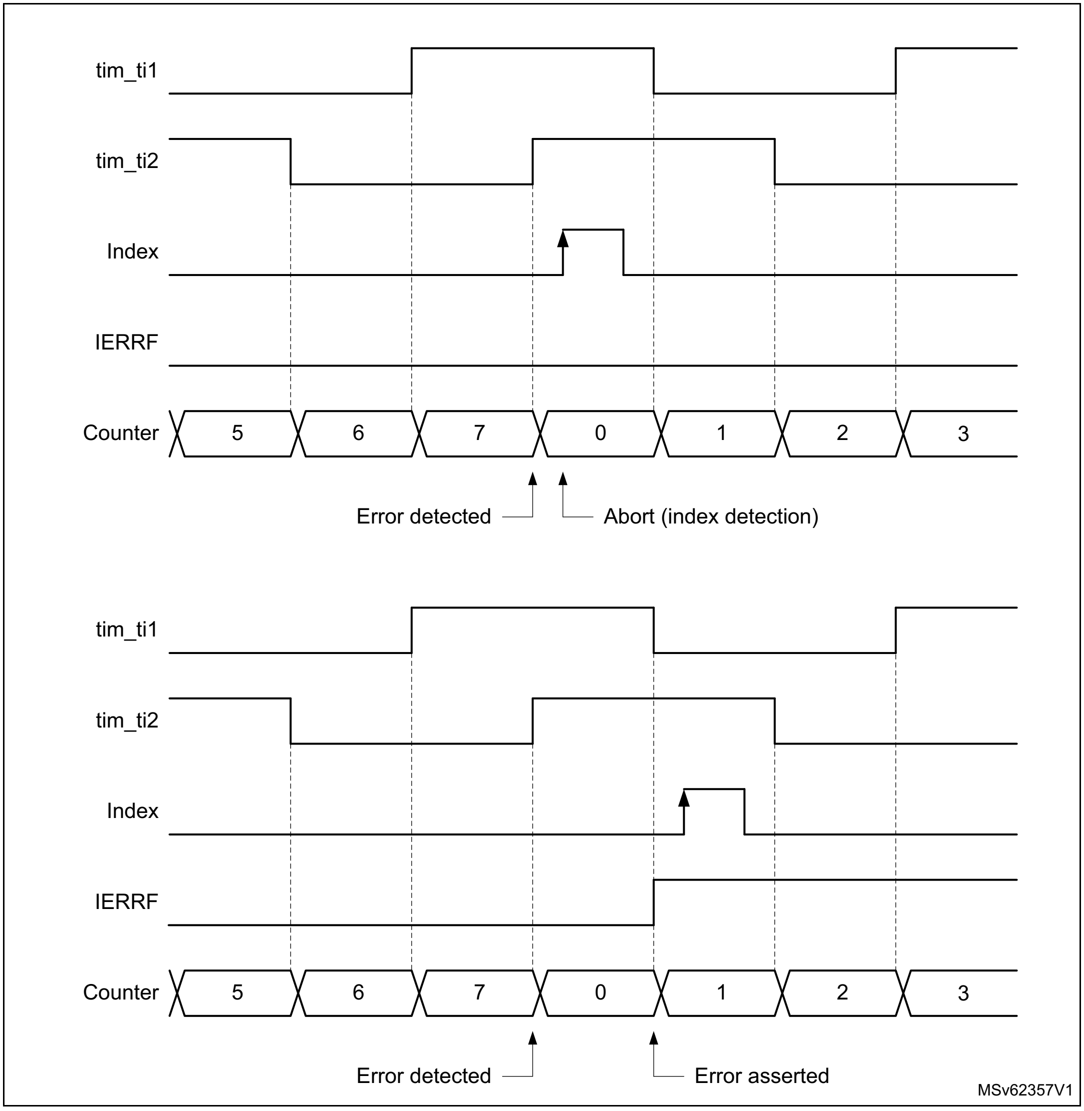
For encoder having an Index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The Index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an Index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

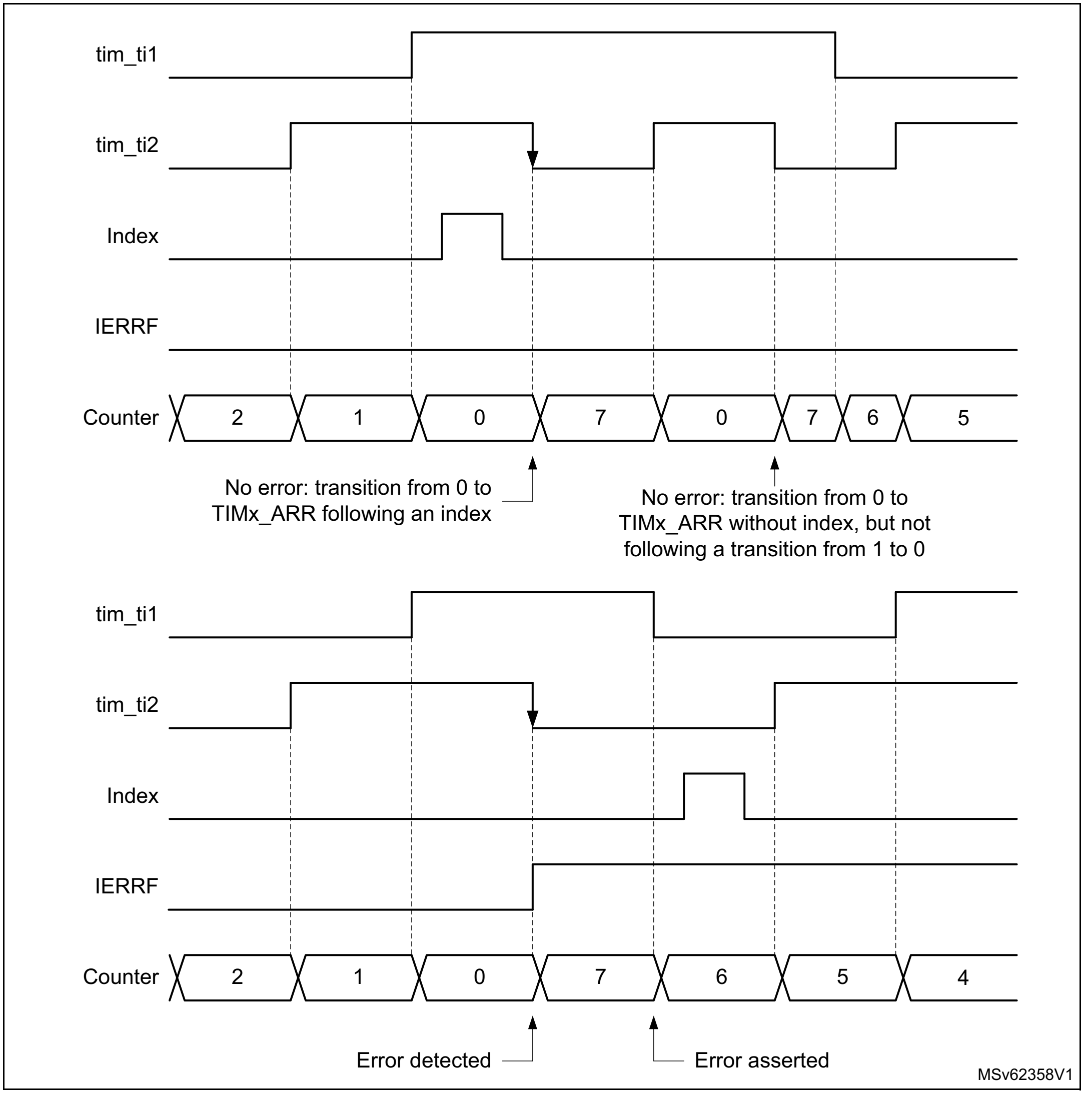
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 349* below.

Figure 349. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 350* below, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 350. Down-counting encode error detection


tim_ti1
tim_ti2
Index
IERRF
Counter
2
1
0
7
0
7
6
5
No error: transition from 0 to TIMx_ARR following an index
No error: transition from 0 to TIMx_ARR without index, but not following a transition from 1 to 0
tim_ti1
tim_ti2
Index
IERRF
Counter
2
1
0
7
6
5
4
Error detected
Error asserted
MSv62358V1

An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder Interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the Index event sets the IDXF interrupt flag in the TIMx_SR status register. An Index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

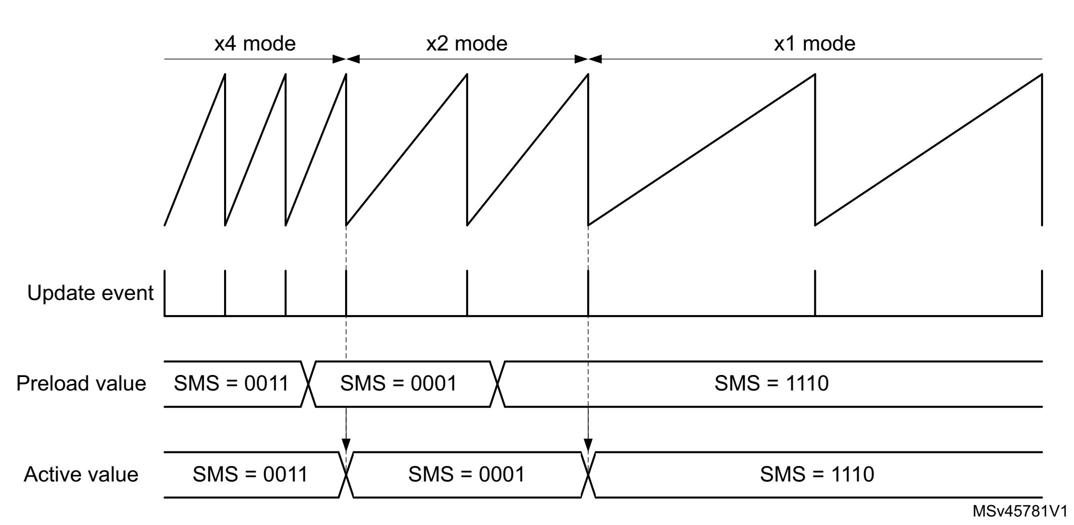
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the Update interrupt rate, by switching from x4 to x2 to x1 mode, as show on the *Figure 351* below.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting. This mode must be used only when index is disabled (bit IE = 0)
- SMSPS = 1: the transfer is triggered by the Index event

Figure 351. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

28.3.26 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bit field to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

28.3.27 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).

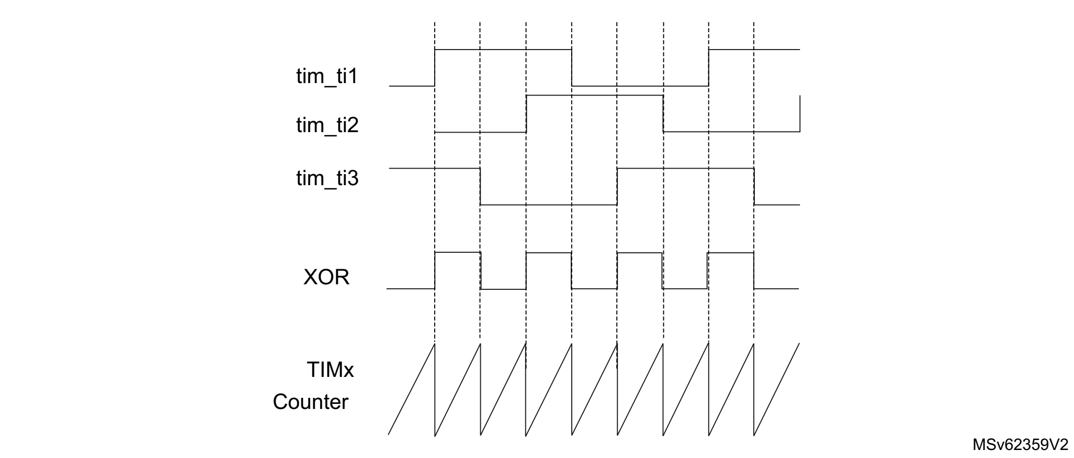
There is no latency between the UIF and UIFCPY flags assertion.

28.3.28 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 tim_ti1, tim_ti2 and tim_ti3.

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 352* below.

Figure 352. Measuring time interval between edges on 3 signals



28.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as "interfacing timer" in *Figure 353*. The "interfacing timer" captures the 3 timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_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 tim_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 tim_trc (See *Figure 296: Capture/compare channel (example: channel 1* *input stage) on page 1113*). 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 (by triggering a COM event). The advanced-control 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 through the tim_trgo output.

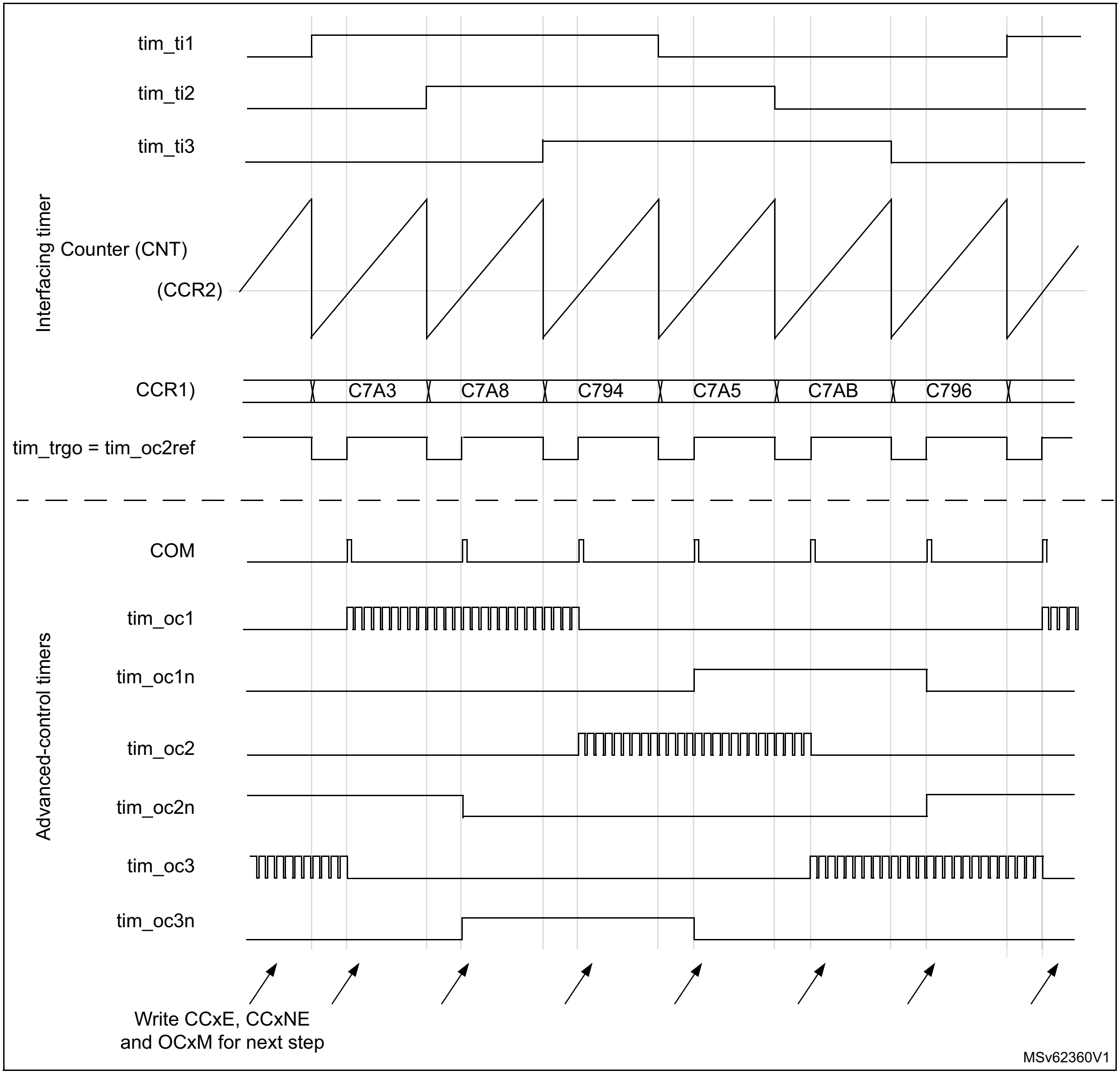
Example: one wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the tim_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 tim_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 (tim_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 tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer, the right tim_itrx 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 tim_oc2ref).

The *Figure 353* describes this example.

Figure 353. Example of Hall sensor interface



28.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 29.4.23: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

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.

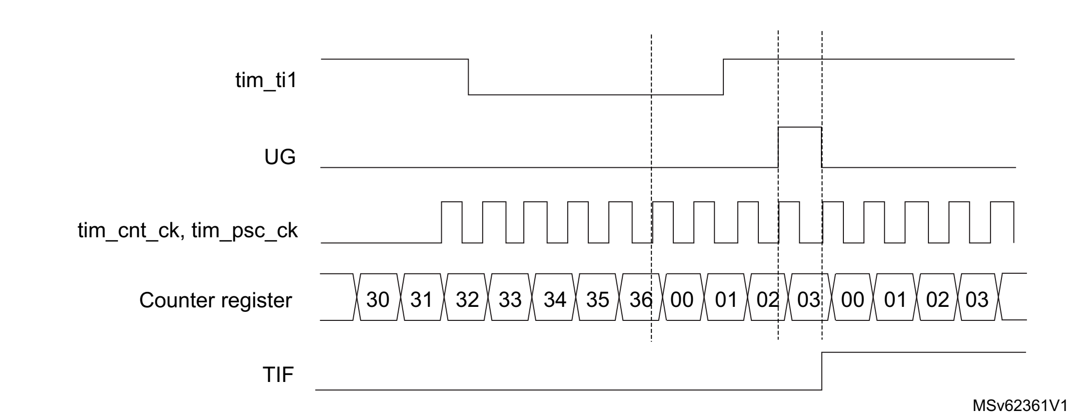
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_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 tim_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 tim_ti1 rising edge. When tim_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 tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 354. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

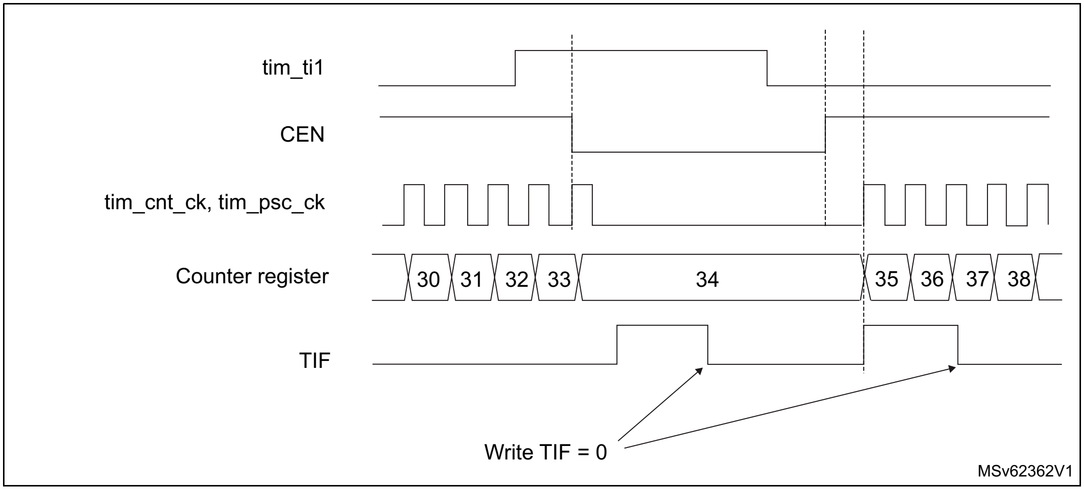
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_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 tim_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 tim_ti1 is low and stops as soon as tim_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 tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 355. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

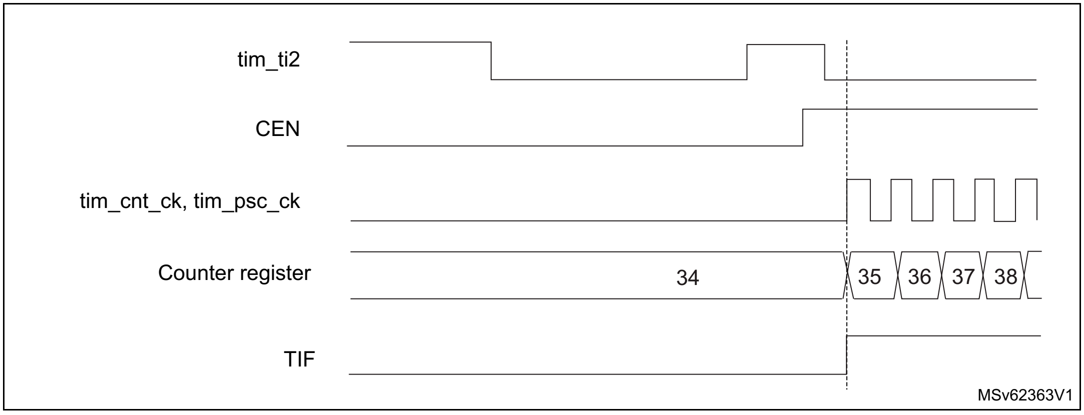
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_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 tim_ti2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 356. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode allows to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

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 tim_etr_in 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 tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

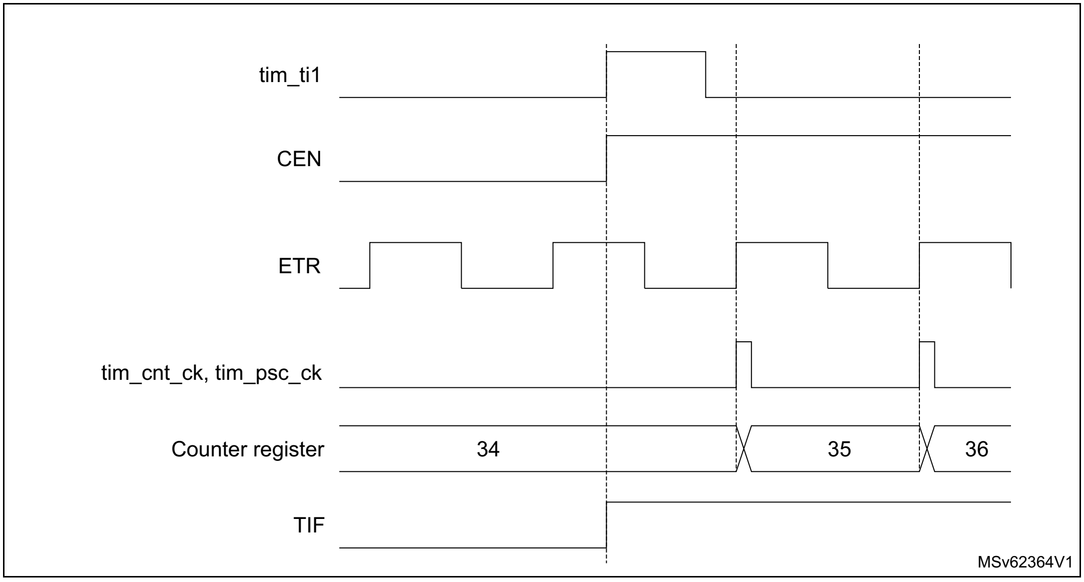
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

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 tim_etr_in 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 tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 357. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_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.

28.3.31 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 tim_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 311 on page 1130*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_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.

28.3.32 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 register 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.

28.3.33 TIM1/TIM8/TIM20 DMA requests

The TIM1/TIM8/TIM20 can generate a DMA request, as shown in *Table 262*.

Table 262. DMA request

DMA acronym	DMA request	Enable control bit
TIM_UP	Update	UDE
TIM_CH1	Capture/compare 1	CC1DE
TIM_CH2	Capture/compare 2	CC2DE
TIM_CH3	Capture/compare 3	CC3DE
TIM_CH4	Capture/compare 4	CC4DE
TIM_COM	Commutation (COM)	COMDE
TIM_TRIG	Trigger	TDE

28.3.34 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIMx_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) 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).

28.4 TIM1/TIM8/TIM20 low-power modes

Table 263. Effect of low-power modes on TIM1/TIM8/TIM20

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

28.5 TIM1/TIM8/TIM20 interrupts

The TIM1/TIM8/TIM20 can generate multiple interrupts, as shown in *Table 264*.

Table 264. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRG_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
	Break2	B2IF		write 0 in B2IF	Yes	No
	System Break	SBIF		write 0 in SBIF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

28.6 TIM1/TIM8/TIM20 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

28.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8, 20)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITH EN	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

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 (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),

00: t_{DTS}=$t_{tim\_ker\_ck}$
01: t_{DTS}=2*$t_{tim\_ker\_ck}$
10: t_{DTS}=4*$t_{tim\_ker\_ck}$
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 6:5 **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.

28.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8, 20)

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.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
						rw		rw	rw	rw	rw		rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	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		rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 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 (tim_trgo2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_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 tim_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 (tim_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 (tim_trgo2).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

1000: **Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

1001: **Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

1010: **Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

1011: **Compare Pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

1100: **Compare Pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1101: **Compare Pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

1110: **Compare Pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1111: **Compare Pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_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 (tim_oc6 output)

Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)

Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)

Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)

Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)

Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)

Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)

Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)

Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)

0: tim_oc1n=0 after a dead-time when MOE=0
1: tim_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 (tim_oc1 output)

0: tim_oc1=0 (after a dead-time) when MOE=0
1: tim_oc1=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 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15..0] multiplexer output is connected to tim_ti1 input

1: tim_ti1_in[15..0], tim_ti2_in[15..0] and tim_ti3_in[15..0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_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 tim_trgo, 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 (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **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. (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

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 tim_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 tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

28.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8, 20)

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.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			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 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

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

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or tim_$\overline{\text{etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge.

1: tim_etr_in 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 tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_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, tim_trgi must not be connected to tim_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 tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_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: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, 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 (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External Trigger input (tim_etrf)
01000: Internal Trigger 0 (tim_itr4)
01001: Internal Trigger 1 (tim_itr5)
01010: Internal Trigger 1 (tim_itr6)
01011: Internal Trigger 1 (tim_itr7)
01100: Internal Trigger 1 (tim_itr8)
01101: Internal Trigger 1 (tim_itr9)
01110: Internal Trigger 1 (tim_itr10)
others: Reserved

See *Table 250: TIMx internal trigger connection* for more details on tim_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: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Bits 16, 2:0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_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: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_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 tim_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 (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=00100). Indeed, tim_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 tim_trgo or the tim_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.

28.6.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8, 20)

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.	TERRIE	IERRIE	DIRIE	IDXIE	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.	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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable

0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable

0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable

0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable

0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 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

28.6.5 TIMx status register (TIMx_SR)(x = 1, 8, 20)

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.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF
								rc_w0	rc_w0	rc_w0	rc_w0			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 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag

This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to ‘0’.

0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag

This flag is set by hardware when an index error is detected. It is cleared by software by writing it to ‘0’.

0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag

This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to ‘0’.

0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag

This flag is set by hardware when an index event is detected. It is cleared by software by writing it to ‘0’.

0: No index event occurred.
1: An index event has occurred

Bits 19:18 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 tim_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 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 downcounting 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 28.6.3: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8, 20)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

28.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8, 20)

Address offset: 0x014

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 to update CCxE, CCxNE and OCxM bits

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. 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).

28.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8, 20)

Address offset: 0x018

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).

31	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

Input capture mode:

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, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_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 tim_ti1 input and the length of the digital filter applied to tim_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: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, 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 (tim_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, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_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).

28.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8, 20)

Address offset: 0x018

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).

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

Output compare mode:

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

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, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_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: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_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. tim_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. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_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 (tim_oc1ref=‘0’) as long as TIMx_CNT>TIMx_CCR1 else active (tim_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 tim_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 tim_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 tim_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 tim_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 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_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.

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).

The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

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, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_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).

28.6.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8, 20)

Address offset: 0x01C

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 3 in input capture mode and channel 4 in 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.	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

Input capture mode

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, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_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, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_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).

28.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8, 20)

Address offset: 0x01C

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 3 in input capture mode and channel 4 in 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

Output compare mode

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 OC3M[3:0] bit description

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, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_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

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT=TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In upcounting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref='0') as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref='1').

0111: PWM mode 2 - In upcounting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In downcounting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_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 tim_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 tim_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 tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref 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.

On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

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, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_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).

28.6.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8, 20)

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.	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	CC4NE	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	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 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

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: tim_oc1n active high.

1: tim_oc1n active low.

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_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).

Note: 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 - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: 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).

Note: 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 265* 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 265. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) tim_ocx=0, tim_ocxn=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) tim_ocx=0	tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref xor CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_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) tim_ocx=CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref x or CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_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: tim_ocx=CCxP, tim_ocxn=CCxNP (if tim_brk or tim_brk2 is triggered).	
			1	0		
			1	1	Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx=OISx and tim_ocxn=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration). *Note: tim_brk2 can only be used if OSSI = OSSR = 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 tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

28.6.12 TIMx counter (TIMx_CNT)(x = 1, 8, 20)

Address offset: 0x024

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

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

28.6.13 TIMx prescaler (TIMx_PSC)(x = 1, 8, 20)

Address offset: 0x028

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 ($f_{tim\_cnt\_ck}$) is equal to $f_{tim\_psc\_ck}$ / (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”).

28.6.14 TIMx auto-reload register (TIMx_ARR)(x = 1, 8, 20)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	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.	ARR[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 28.3.3: Time-base unit on page 1093* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

28.6.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8, 20)

Address offset: 0x030

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 reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.
When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, 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.

28.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8, 20)

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.	CCR1[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: 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 tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

28.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8, 20)

Address offset: 0x038

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.	CCR2[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: 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 tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

28.6.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8, 20)

Address offset: 0x03C

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.	CCR3[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: 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 signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

28.6.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8, 20)

Address offset: 0x040

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.	CCR4[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: 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 tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

28.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8, 20)

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.	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 BKBID/BK2BID/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 tim_brk in input mode

1: Break input tim_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 tim_brk is armed

1: Break input tim_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 tim_brk2 is active low

1: Break input tim_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

This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 316: Break and Break2 circuitry overview*).

0: Break2 function disabled

1: Break2 function enabled

Note: The BRKIN2 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 tim_brk2 input and the length of the digital filter applied to tim_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, tim_brk2 acts asynchronously
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, 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 tim_brk input and the length of the digital filter applied to tim_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, tim_brk acts asynchronously
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, 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 (tim_brk or tim_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 28.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8, 20)*).

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 tim_brk and tim_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 tim_brk is active low
1: Break input tim_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 316: 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 28.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8, 20)*).

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 28.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8, 20)*).

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 BKBID/BK2BID/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])xt_{dtg} with T_{dtg}=2xt_{DTS}.
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with T_{dtg}=8xt_{DTS}.
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with T_{dtg}=16xt_{DTS}.
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us 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).

28.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8, 20)

Address offset: 0x048

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.	CCR5[19:16]			
rw	rw	rw										rw	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 tim_oc5ref on tim_oc3refc
1: tim_oc3refc is the logical AND of tim_oc3ref and tim_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 tim_oc5ref on tim_oc2refc
1: tim_oc2refc is the logical AND of tim_oc2ref and tim_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 oc1refc
1: oc1refc is the logical AND of oc1ref 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 28:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: 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 tim_oc5 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

28.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8, 20)

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.	CCR6[19:16]			
												rw	rw	rw	rw
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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR6[19:0]**: 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 tim_oc6 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

28.6.23 TIMx capture/compare mode register 3 (TIMx_CCMR3) (x = 1, 8, 20)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

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

Bits 24, 14:12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

Bits 16, 6:4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, must be kept at reset value.

28.6.24 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8, 20)

Address offset: 0x054

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.	DTPE	DTAE
														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.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

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 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

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 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x t_{dtg} with t_{dtg}=t_{DTS}.
DTGF[7:5]=10x => DTF=(64+DTGF[5:0])xt_{dtg} with T_{dtg}=2xt_{DTS}.
DTGF[7:5]=110 => DTF=(32+DTGF[4:0])xt_{dtg} with T_{dtg}=8xt_{DTS}.
DTGF[7:5]=111 => DTF=(32+DTGF[4:0])xt_{dtg} with T_{dtg}=16xt_{DTS}.
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us 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).

28.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8, 20)

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.	PWPRSC[2:0]			PW[7:0]							
					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.	IPOS[1:0]		FIDX	Res.	Res.	IDIR[1:0]		IE
								rw	rw	rw			rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])})$ x $t_{tim\_ker\_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0]$ x t_{PWG}

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Note: The IDR[1:0] bitfield must be written when IE bit is reset (index disabled).

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

28.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8, 20)

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.	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 tim_ti4[0..15] input

0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[0..15] input

0000: tim_ti3_in0: TIMx_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[0..15] input

0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[0..15] input

0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

28.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8, 20)

Address offset: 0x060

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]		BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
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 17:14 **ETRSEL[3:0]**: etr_in source selection

These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp4 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 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp3 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 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp2 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**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp1 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**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: TIMx_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).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer’s tim_brk input. tim_brk_cmp8 output is ‘ORed’ with the other tim_brk sources.

0: tim_brk_cmp8 input disabled
1: tim_brk_cmp8 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 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer’s tim_brk input. tim_brk_cmp7 output is ‘ORed’ with the other tim_brk sources.

0: tim_brk_cmp7 input disabled
1: tim_brk_cmp7 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 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer’s tim_brk input. tim_brk_cmp6 output is ‘ORed’ with the other tim_brk sources.

0: tim_brk_cmp6 input disabled
1: tim_brk_cmp6 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 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 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 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 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 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 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 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 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**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 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**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_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 Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals for product specific implementation.

28.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8, 20)

Address offset: 0x064

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.	OCRSEL[2:0]		
													rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
...
111: tim_ocref_clr7
Refer to *Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 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 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 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 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 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**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 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**: TIMx_BKIN2 input polarity

This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIMx_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIMx_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).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer’s tim_brk2 input. tim_brk2_cmp8 output is ‘ORed’ with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 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 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer’s tim_brk2 input. tim_brk2_cmp7 output is ‘ORed’ with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 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 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer’s tim_brk2 input. tim_brk2_cmp6 output is ‘ORed’ with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 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 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer’s tim_brk2 input. tim_brk2_cmp5 output is ‘ORed’ with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 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 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer’s tim_brk2 input. tim_brk2_cmp4 output is ‘ORed’ with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 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 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 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 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 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**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 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**: TIMx_BKIN2 input enable

This bit enables the TIMx_BKIN2 alternate function input for the timer's tim_brk2 input. TIMx_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIMx_BKIN2 input disabled

1: TIMx_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 Section 28.3.2: TIM1/TIM8/TIM20 pins and internal signals for product specific implementation.

28.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8, 20)

Address offset: 0x3DC

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.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 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
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_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 are 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 are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are 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
...

28.6.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8, 20)

Address offset: 0x3E0

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).

28.6.31 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 266. TIMx 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	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	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	0
0x004	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	OIS4N	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	0		0
0x008	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETP S [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	0	0
0x00C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	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	0	0	0	0
0x010	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	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	0	0	0	0
0x014	**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.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x018	**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]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	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	CC2 S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	**TIMx_CCMR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4 S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4 S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 266. TIMx 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
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	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	0
0x024	TIMx_CNT	UIFCPY	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
0x028	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
0x02C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	TIMx_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
0x034	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIMx_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIMx_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIMx_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	TIMx_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
0x048	TIMx_CCR5	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:0]																			
	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
0x04C	TIMx_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	TIMx_CCMR3	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		

Table 266. TIMx 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
0x054	**TIMx_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x058	**TIMx_ECR**	Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	Res.	Res.	IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0			0	0	0
0x05C	**TIMx_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
0x060	**TIMx_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]				BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x064	**TIMx_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL [2:0]			Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value														0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	**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
0x3E0	**TIMx_DMAR**	DMAB[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 2.2 on page 81* for the register boundary addresses.

29 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

29.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-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 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.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 29.4.23: Timer synchronization*.

29.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 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

29.3 TIM2/TIM3/TIM4/TIM5 implementation

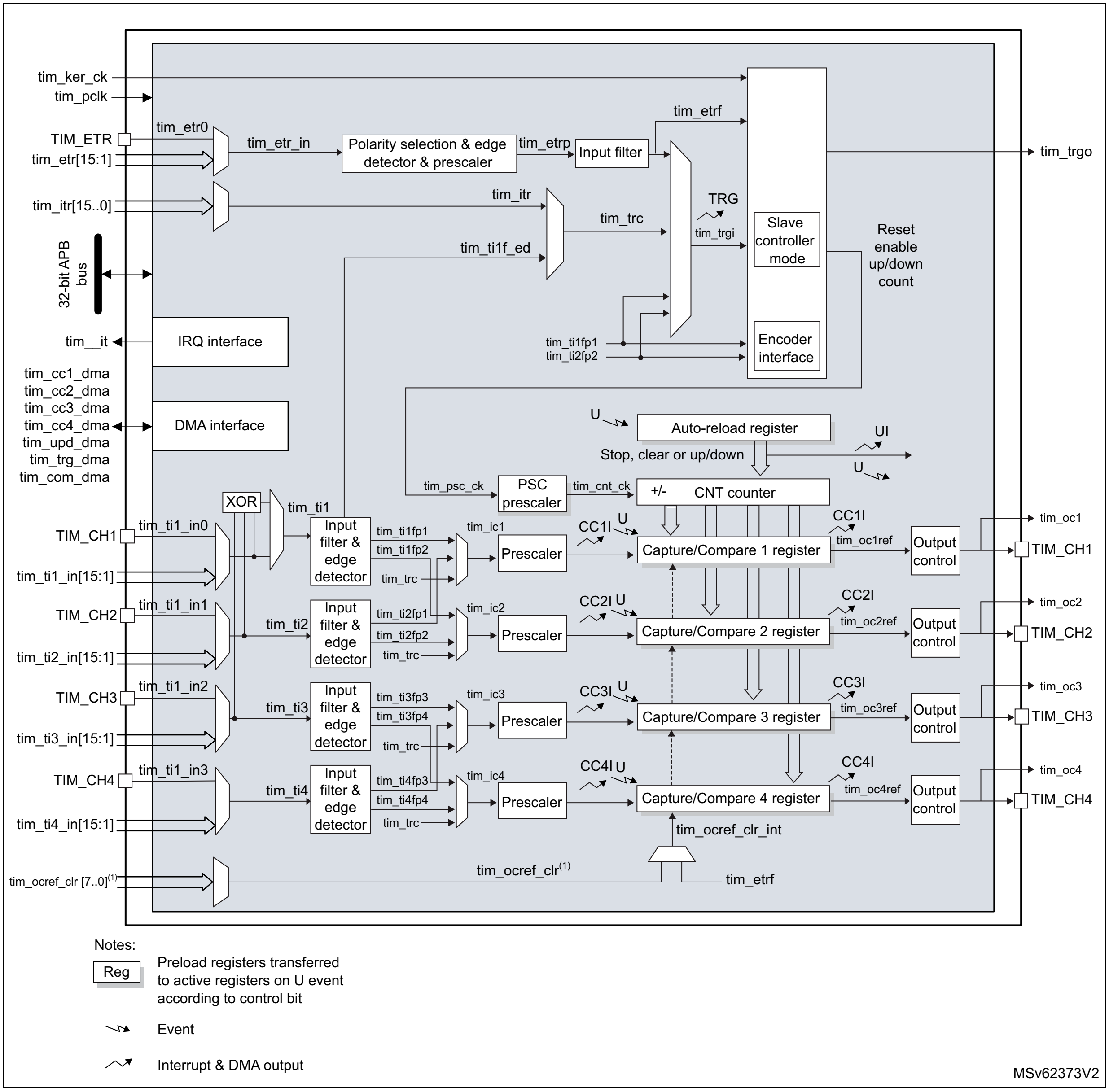
Table 267. STM32G4 Series general purpose timers

Timer instance	TIM2	TIM3	TIM4	TIM5
Resolution	32-bit	16-bit	16-bit	32-bit
OCREF clear selection	Yes	Yes	No	No
Sources	tim_etrf tim_ocref_clr[7:0]	tim_etrf tim_ocref_clr[7:0]	tim_etrf -	tim_etrf -

29.4 TIM2/TIM3/TIM4/TIM5 functional description

29.4.1 Block diagram

Figure 358. General-purpose timer block diagram



1. This feature is not available on all timers, refer to the *Section 29.3: TIM2/TIM3/TIM4/TIM5 implementation*.

29.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 268 and *Table 269* in this section summarize the TIM inputs and outputs.

Table 268. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 269. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 269. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trg_dma	Output	Timer trigger dma request.
tim_com_dma	Output	Timer commutation dma request.

Table 270, *Table 271*, *Table 272* and *Table 273* are listing the sources connected to the tim_ti[1..4] input multiplexers.

Table 270. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti1_in0	TIM2_CH1	TIM3_CH1	TIM4_CH1	TIM5_CH1
tim_ti1_in1	comp1_out	comp1_out	comp1_out	LSI
tim_ti1_in2	comp2_out	comp2_out	comp2_out	LSE
tim_ti1_in3	comp3_out	comp3_out	comp3_out	RTC wake-up
tim_ti1_in4	comp4_out	comp4_out	comp4_out	comp1_out
tim_ti1_in5	comp5_out	comp5_out	comp5_out	comp2_out
tim_ti1_in6	Reserved	comp6_out	comp6_out	comp3_out
tim_ti1_in7		comp7_out	comp7_out	comp4_out
tim_ti1_in8		Reserved	Reserved	comp5_out
tim_ti1_in9				comp6_out
tim_ti1_in10				comp7_out
tim_ti1_in[11..15]	Reserved			

Table 271. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in0	TIM2_CH2	TIM3_CH2	TIM4_CH2	TIM5_CH2
tim_ti2_in1	comp1_out	comp1_out	comp1_out	comp1_out
tim_ti2_in2	comp2_out	comp2_out	comp2_out	comp2_out

Table 271. Interconnect to the tim_ti2 input multiplexer (continued)

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in3	comp3_out	comp3_out	comp3_out	comp3_out
tim_ti2_in4	comp4_out	comp4_out	comp4_out	comp4_out
tim_ti2_in5	comp6_out	comp5_out	comp5_out	comp5_out
tim_ti2_in6	Reserved	comp6_out	comp6_out	comp6_out
tim_ti2_in7		comp7_out	comp7_out	comp7_out
tim_ti2_in[8..15]	Reserved			

Table 272. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti3_in0	TIM2_CH3	TIM3_CH3	TIM4_CH3	TIM5_CH3
tim_ti3_in1	comp4_out	comp3_out	comp5_out	Reserved
tim_ti2_in[2..15]	Reserved			

Table 273. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti4_in0	TIM2_CH4	TIM3_CH4	TIM4_CH4	TIM5_CH4
tim_ti4_in1	comp1_out	Reserved	comp6_out	Reserved
tim_ti4_in2	comp2_out		Reserved	
tim_ti4_in[3..15]	Reserved			

Table 274 lists the internal sources connected to the tim_itr input multiplexer.

Table 274. TIMx internal trigger connection

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr0	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_itr1	Reserved	tim2_trgo	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	Reserved	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo	Reserved	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo	tim5_trgo	Reserved
tim_itr5	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_itr6	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo

Table 274. TIMx internal trigger connection (continued)

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr7	tim16_oc1	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr8	tim17_oc1	tim17_oc1	tim17_oc1	tim17_oc1
tim_itr9	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo
tim_itr10	hrtim_out_sync2	hrtim_out_sync2	hrtim_out_sync2	hrtim_out_sync2
tim_itr11	USB SOF SYNC	Reserved	Reserved	Reserved
tim_itr11..15	Reserved			

Table 275 lists the internal sources connected to the tim_etr input multiplexer.

Table 275. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR
tim_etr1	comp1_out	comp1_out	comp1_out	comp1_out
tim_etr2	comp2_out	comp2_out	comp2_out	comp2_out
tim_etr3	comp3_out	comp3_out	comp3_out	comp3_out
tim_etr4	comp4_out	comp4_out	comp4_out	comp4_out
tim_etr5	comp5_out	comp5_out	comp5_out	comp5_out
tim_etr6	comp6_out	comp6_out	comp6_out	comp6_out
tim_etr7	comp7_out	comp7_out	comp7_out	comp7_out
tim_etr8	TIM3_ETR	TIM2_ETR	TIM3_ETR	TIM2_ETR
tim_etr9	TIM4_ETR	TIM4_ETR	TIM5_ETR	TIM3_ETR
tim_etr10	TIM5_ETR	Reserved	Reserved	Reserved
tim_etr11	LSE	adc2_awd1		
tim_etr12	Reserved	adc2_awd2		
tim_etr13		adc2_awd3		
tim_etr[14..15]	Reserved			

Table 276 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 276. Interconnect to the tim_ocref_clr input multiplexer

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr0	comp1_out	comp1_out	Reserved	Reserved
tim_ocref_clr1	comp2_out	comp2_out		
tim_ocref_clr2	comp3_out	comp3_out		
tim_ocref_clr3	comp4_out	comp4_out		
tim_ocref_clr4	comp5_out	comp5_out		
tim_ocref_clr5	comp6_out	comp6_out		
tim_ocref_clr6	comp7_out	comp7_out		
tim_ocref_clr7	Reserved			

29.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-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)

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 tim_cnt_ck, 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 359 and *Figure 360* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 359. Counter timing diagram with prescaler division change from 1 to 2

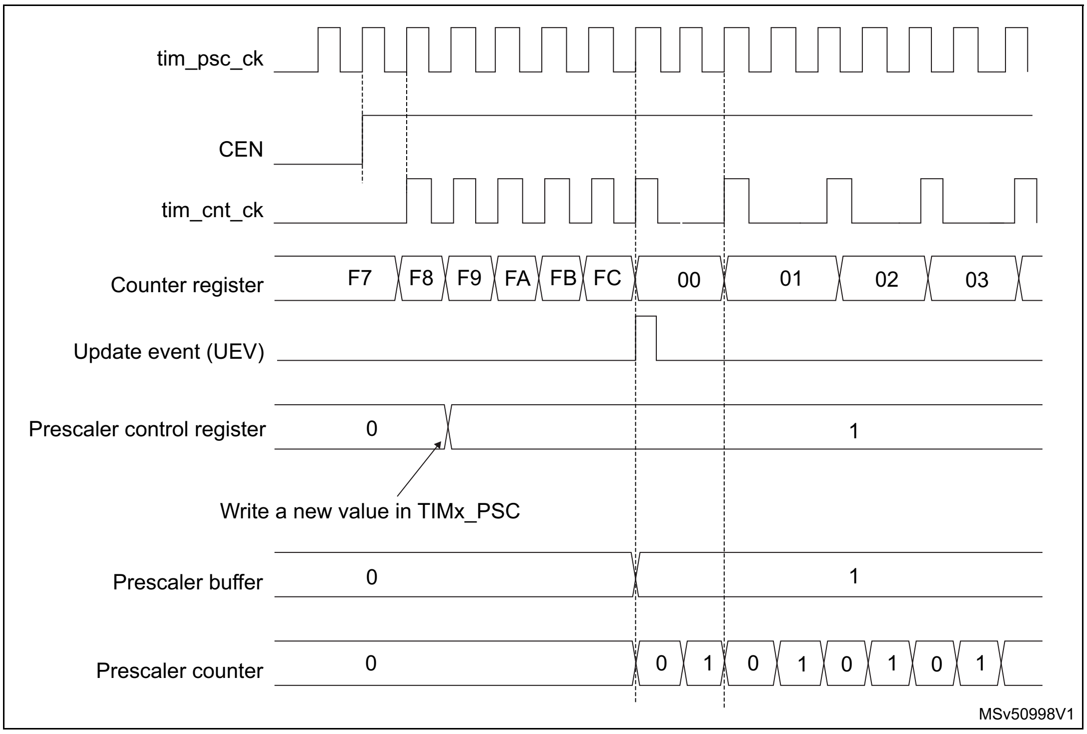
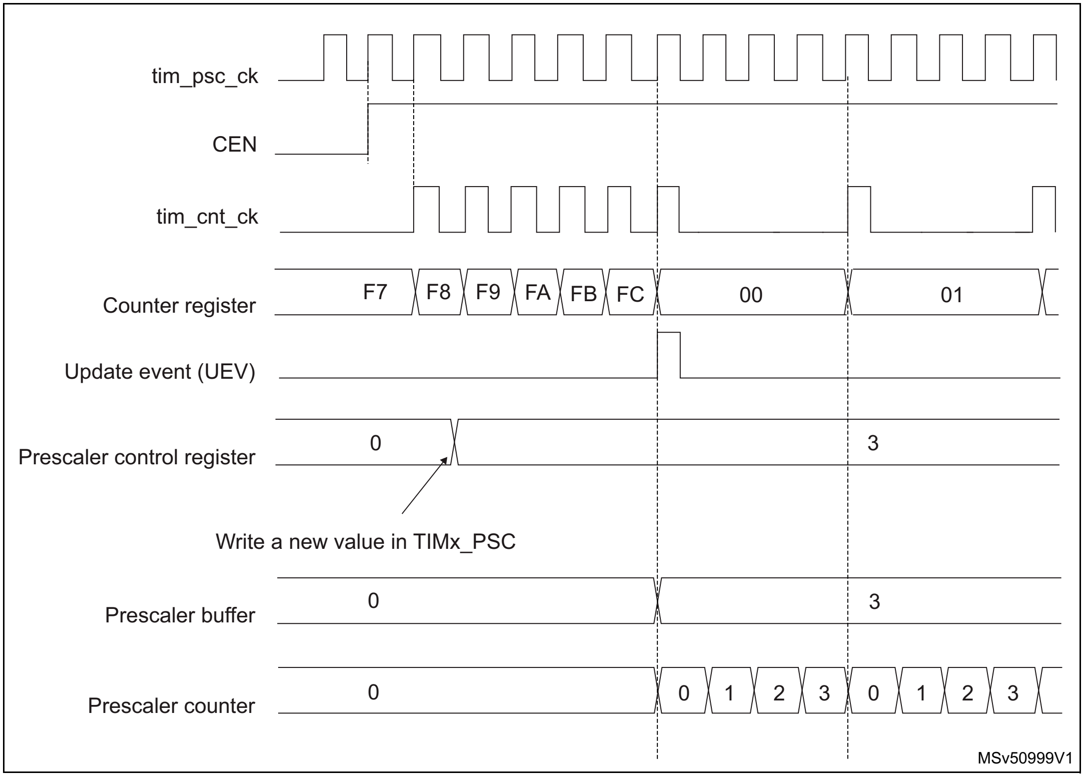


Figure 360. Counter timing diagram with prescaler division change from 1 to 4



29.4.4 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 361. Counter timing diagram, internal clock divided by 1

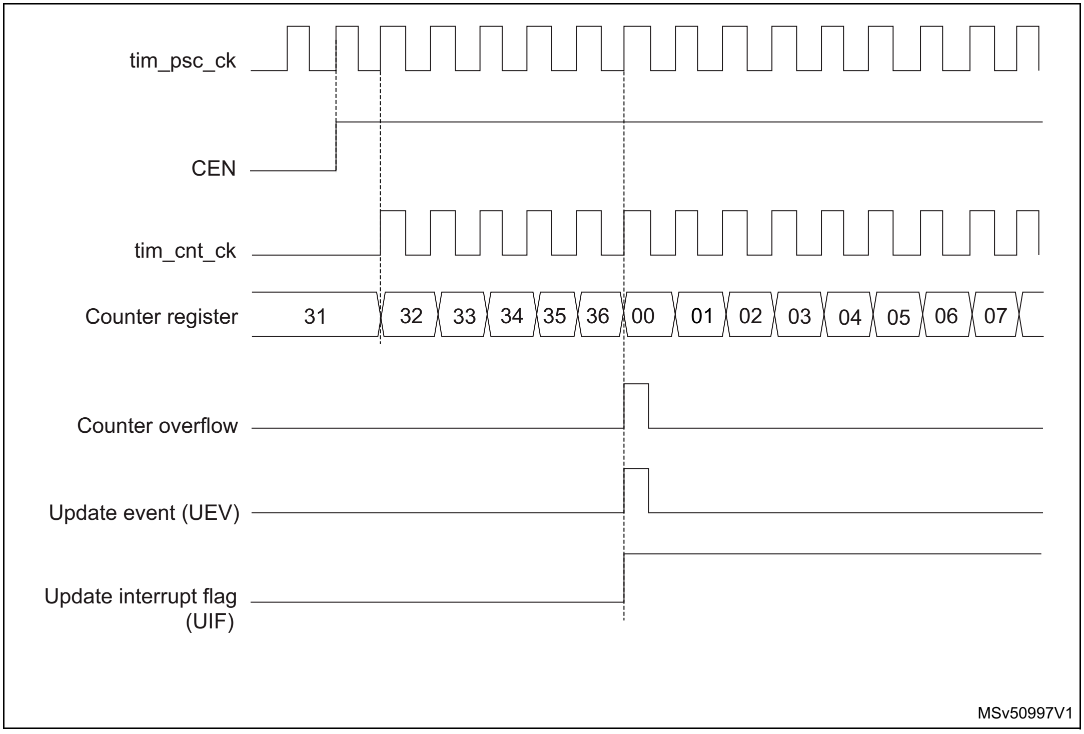


Figure 362. Counter timing diagram, internal clock divided by 2

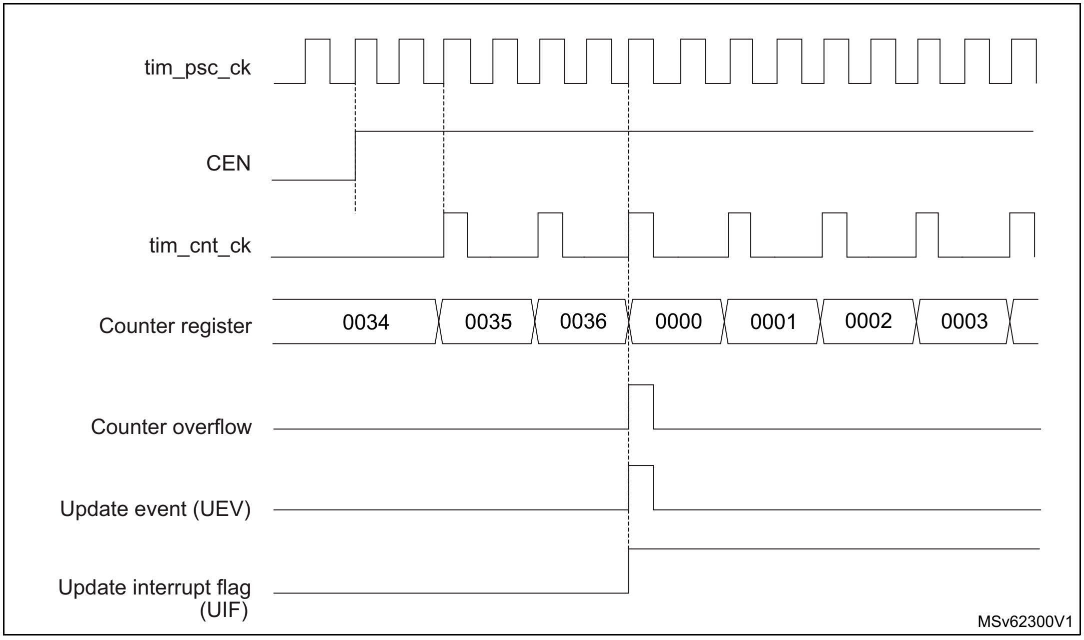


Figure 363. Counter timing diagram, internal clock divided by 4

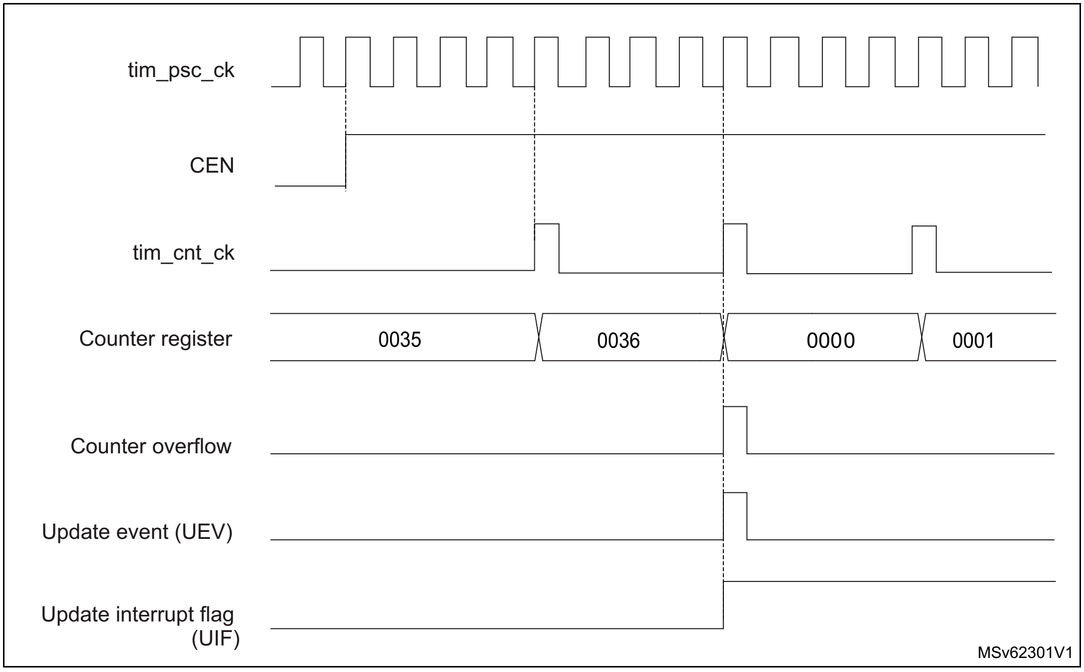


Figure 364. Counter timing diagram, internal clock divided by N

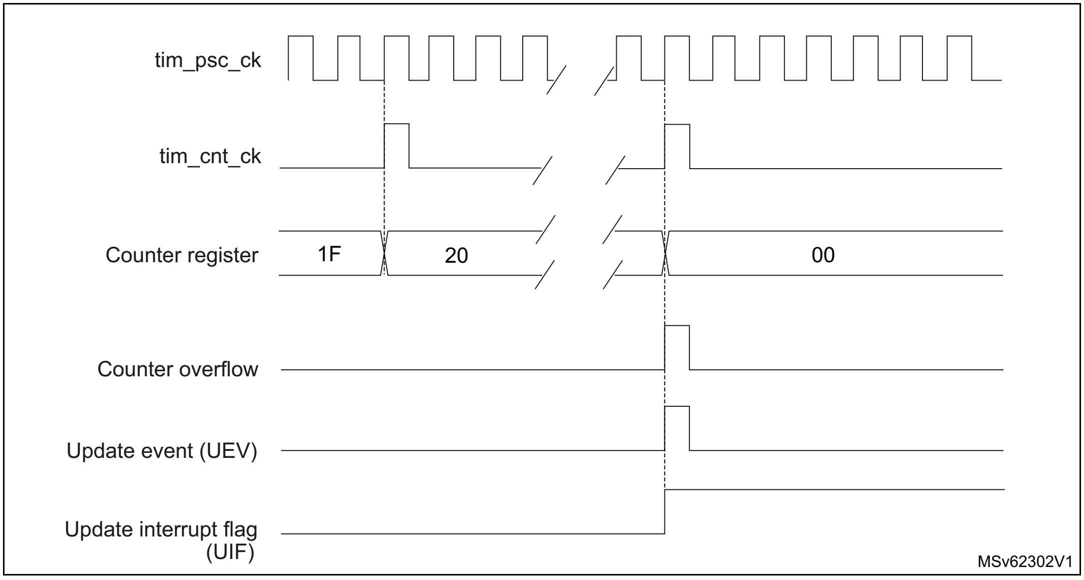


Figure 365. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

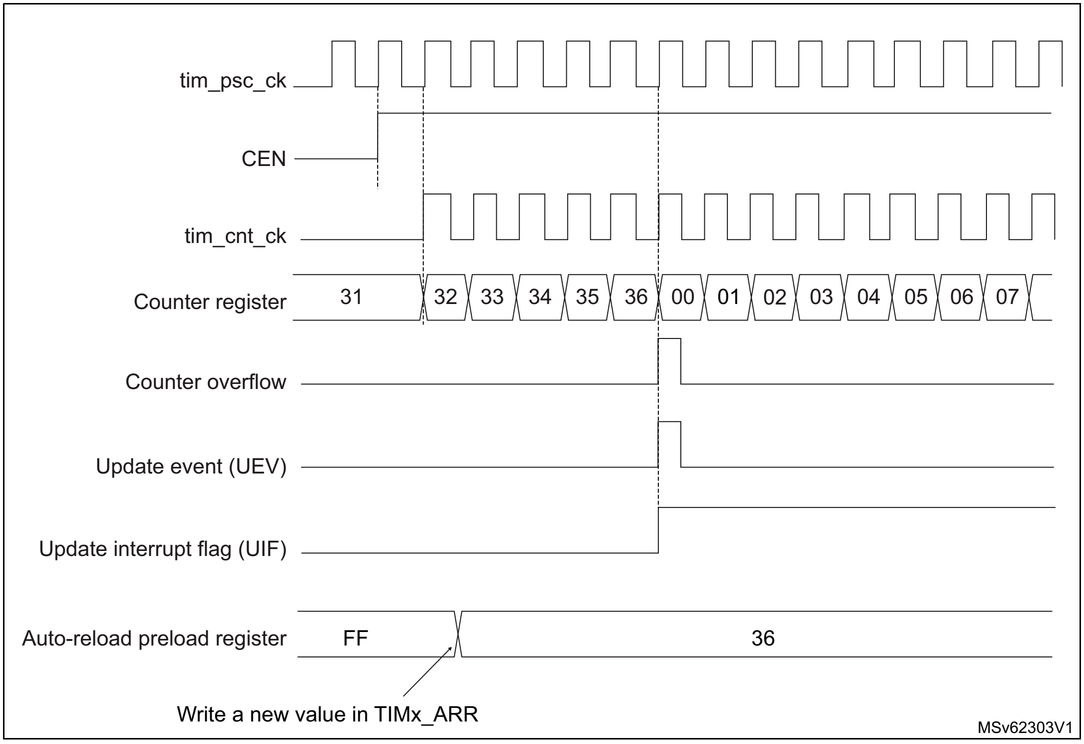
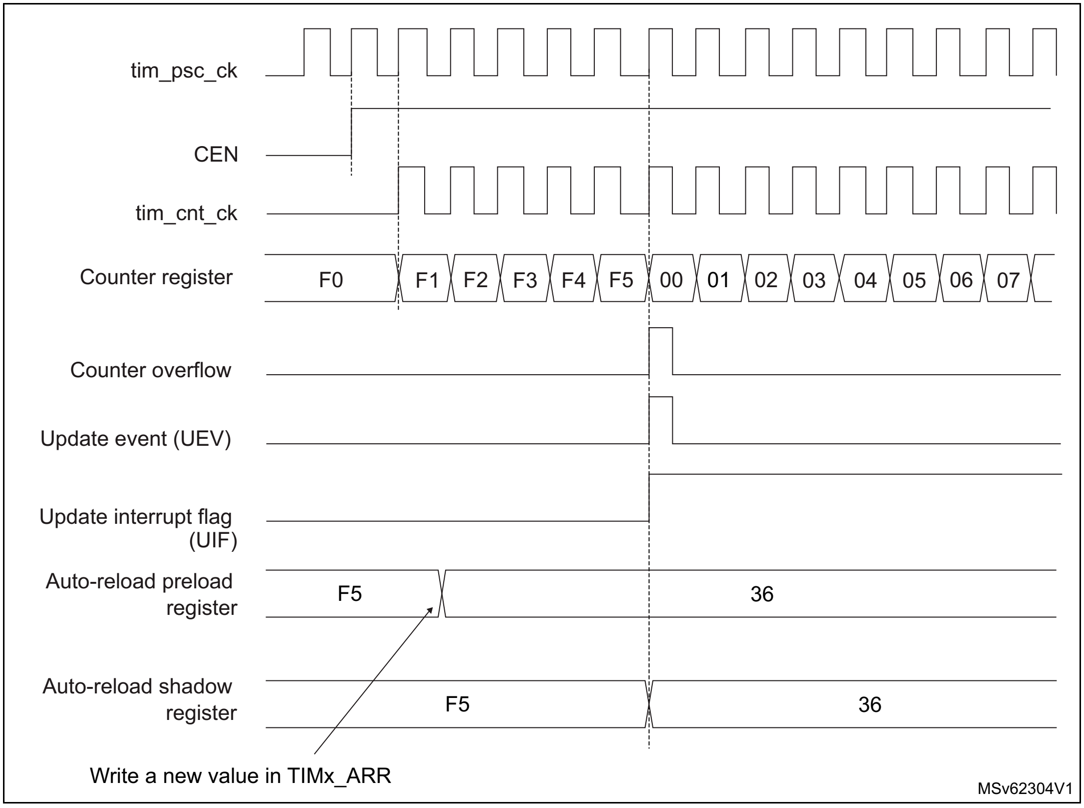


Figure 366. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



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 367. Counter timing diagram, internal clock divided by 1

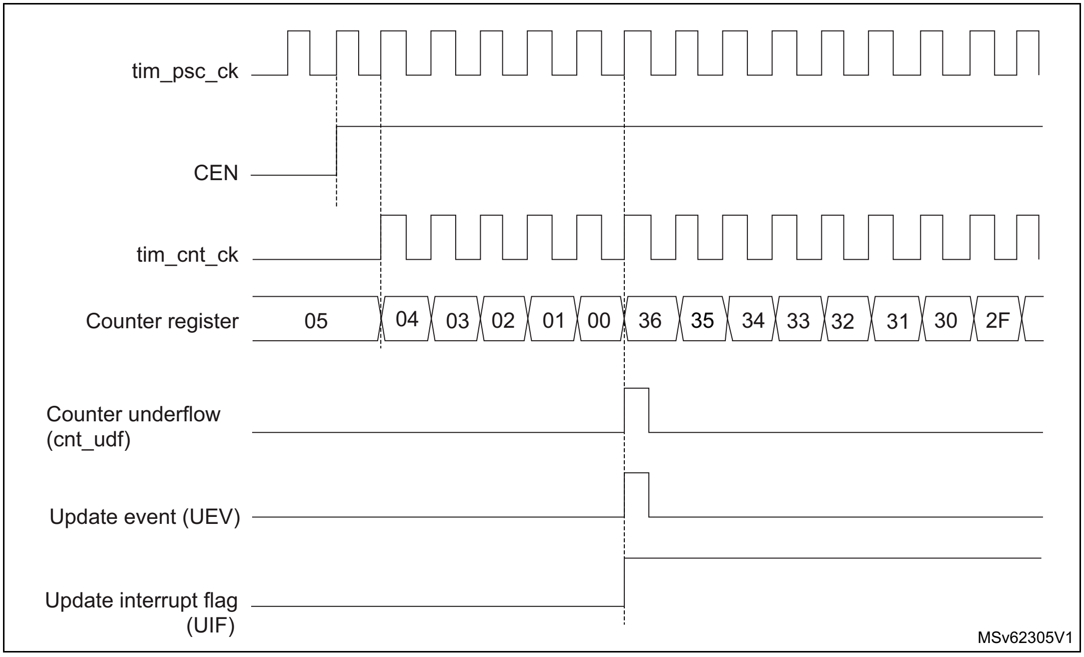


Figure 368. Counter timing diagram, internal clock divided by 2

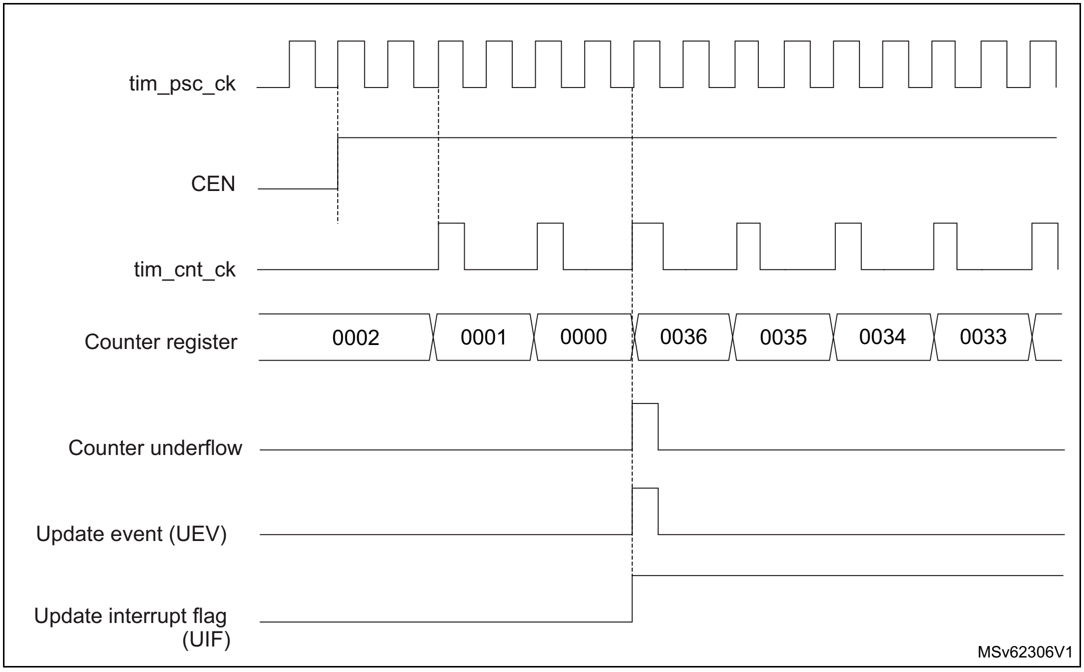


Figure 369. Counter timing diagram, internal clock divided by 4

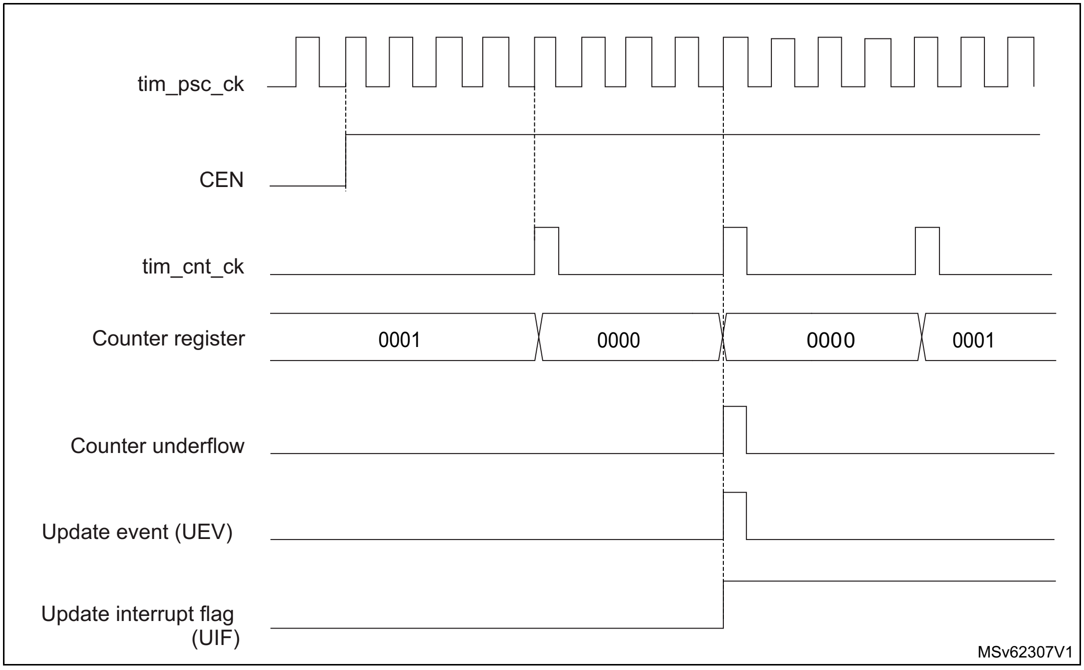


Figure 370. Counter timing diagram, internal clock divided by N

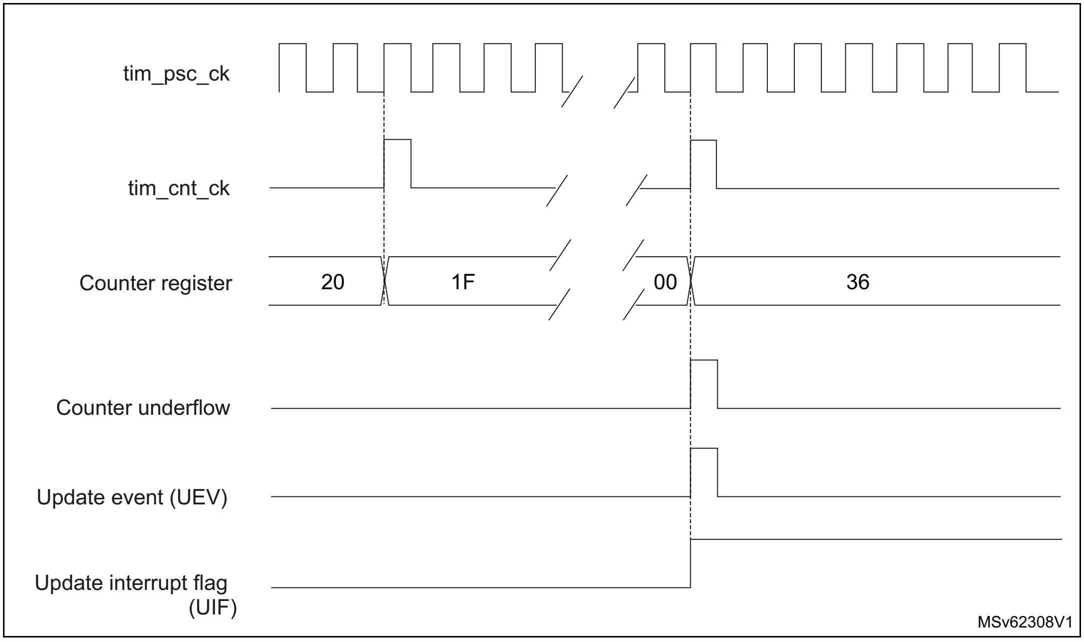
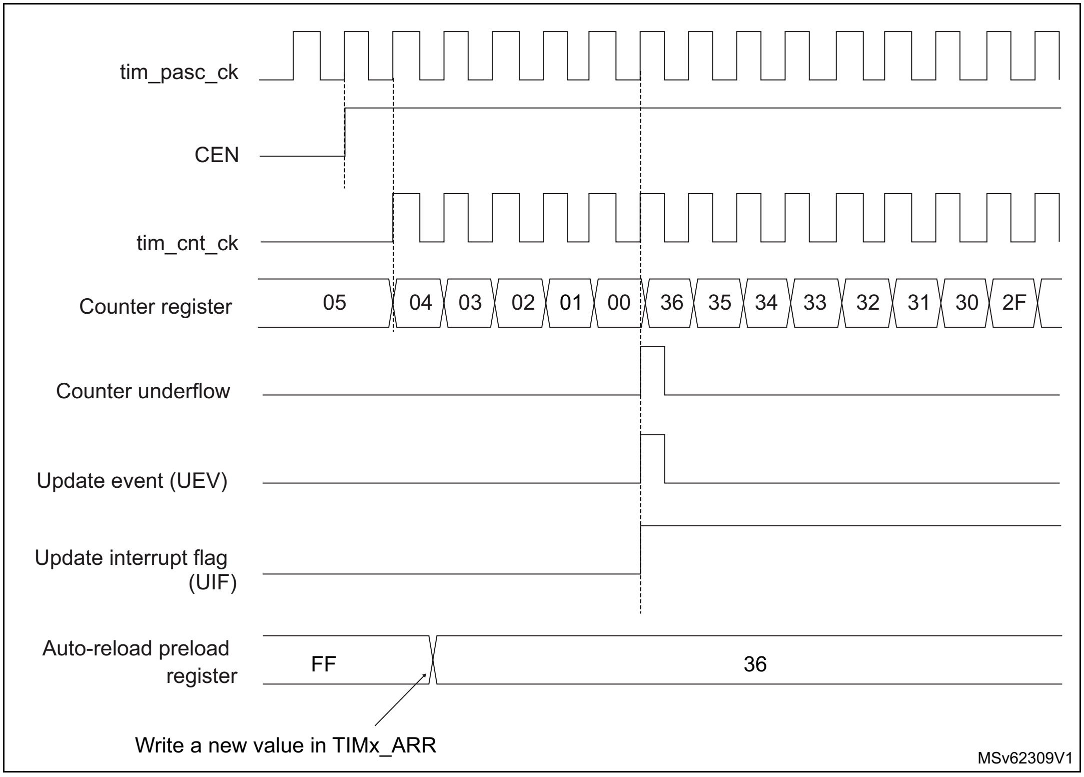


Figure 371. Counter timing diagram, Update event



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.

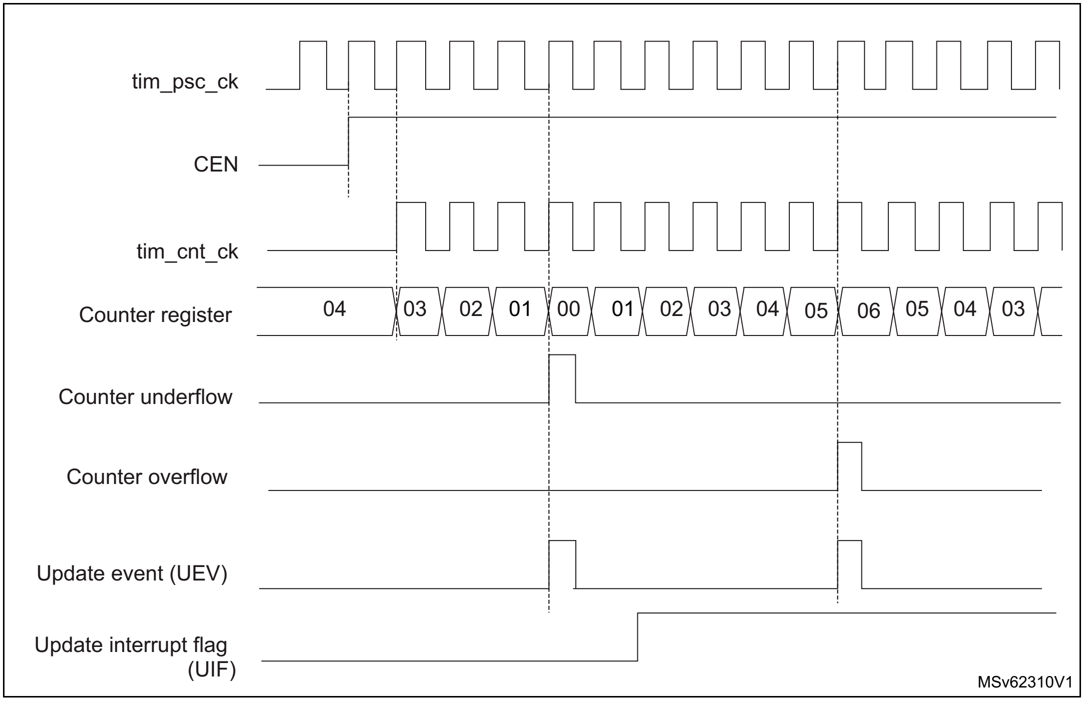
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on 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 372. 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 29.5.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5) on page 1308*).

Figure 373. Counter timing diagram, internal clock divided by 2

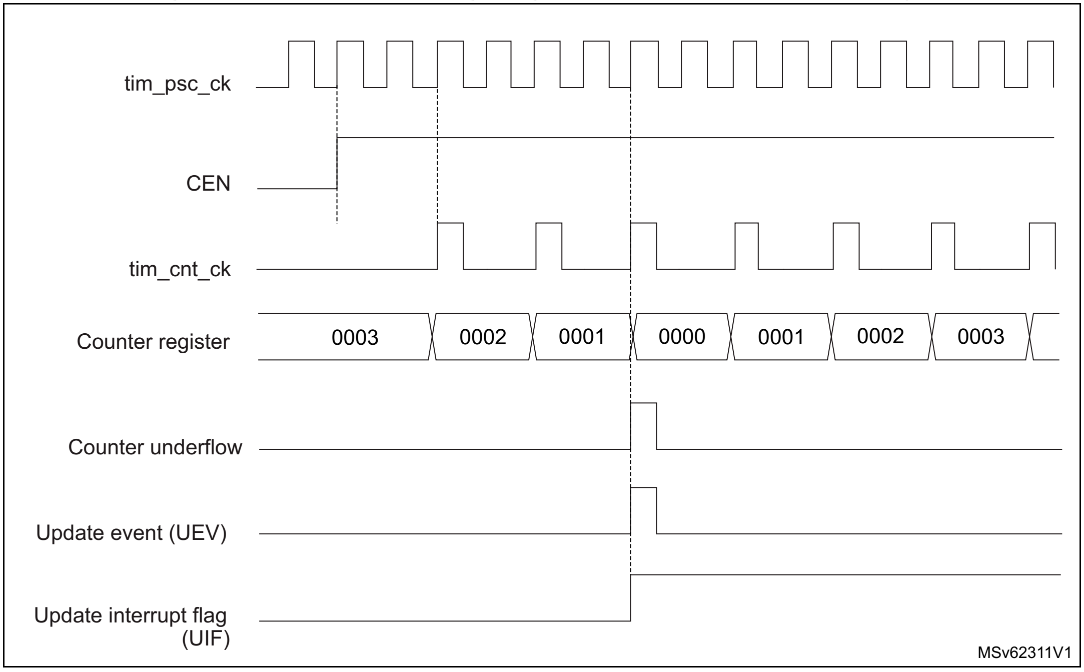
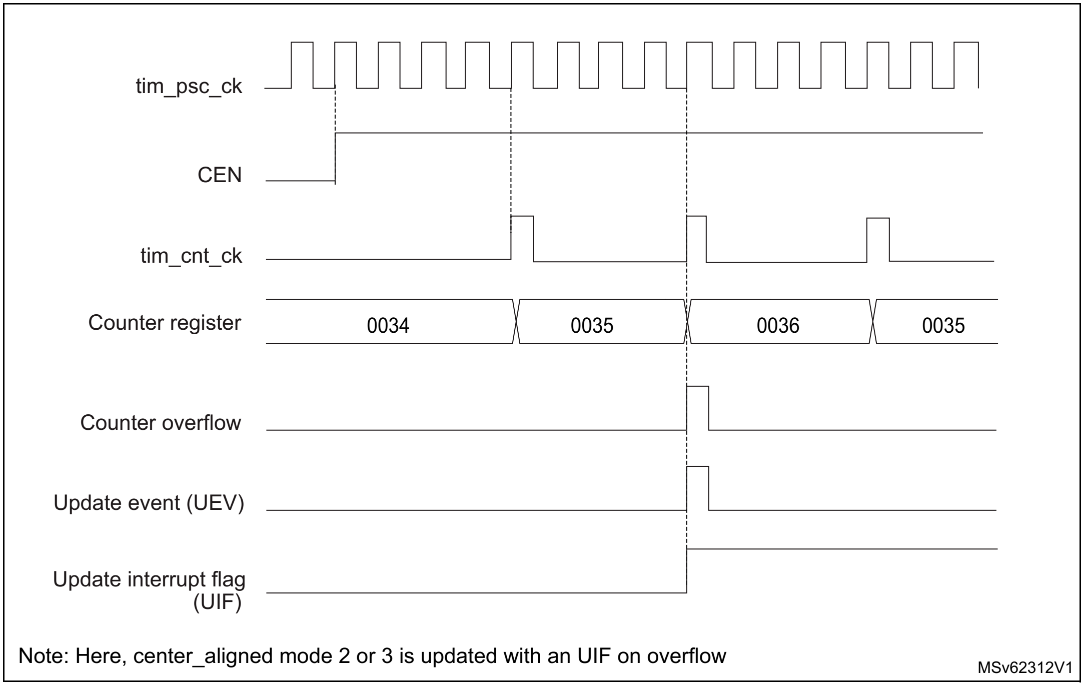


Figure 374. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



Note: Here, center_aligned mode 2 or 3 is updated with an UIF on overflow

1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 375. Counter timing diagram, internal clock divided by N

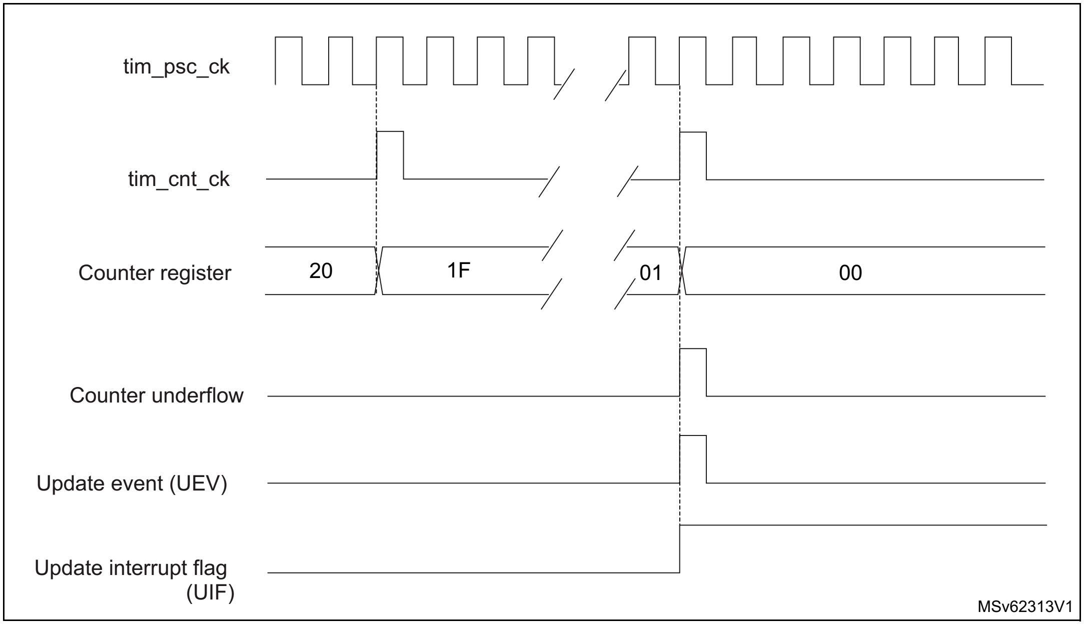


Figure 376. Counter timing diagram, Update event with ARPE=1 (counter underflow)

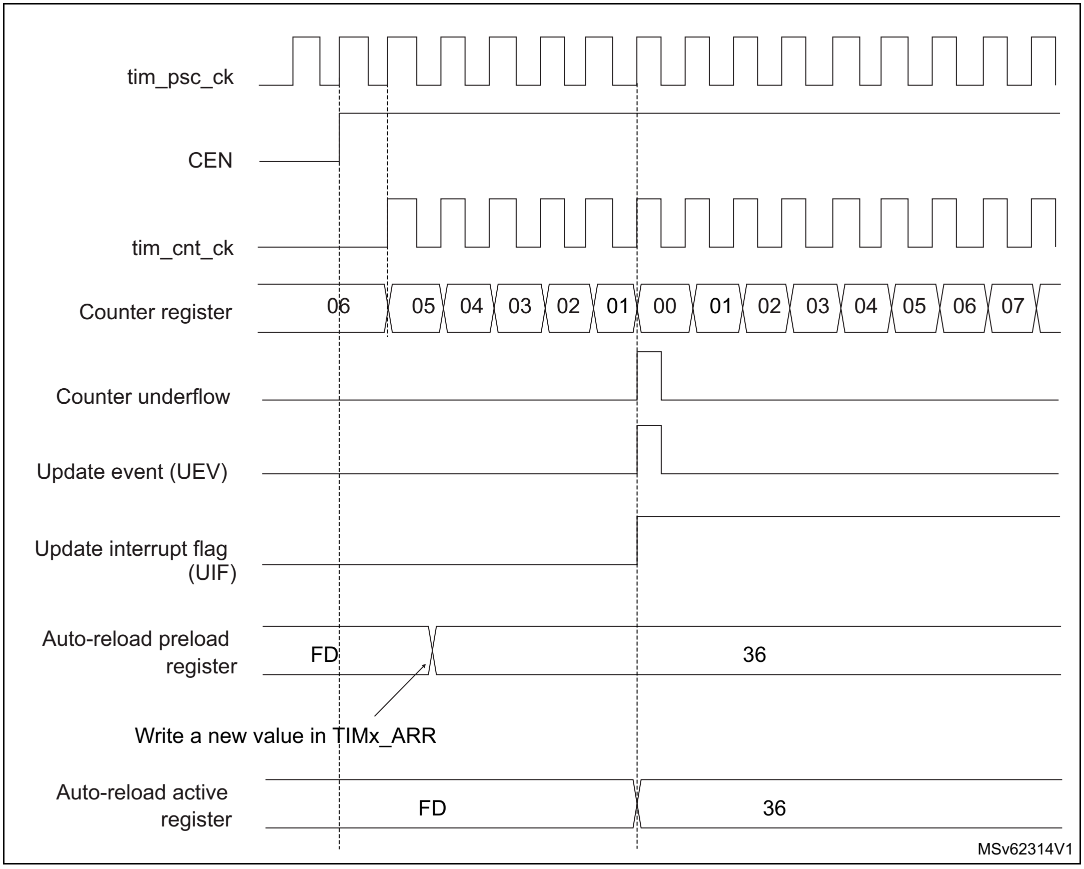
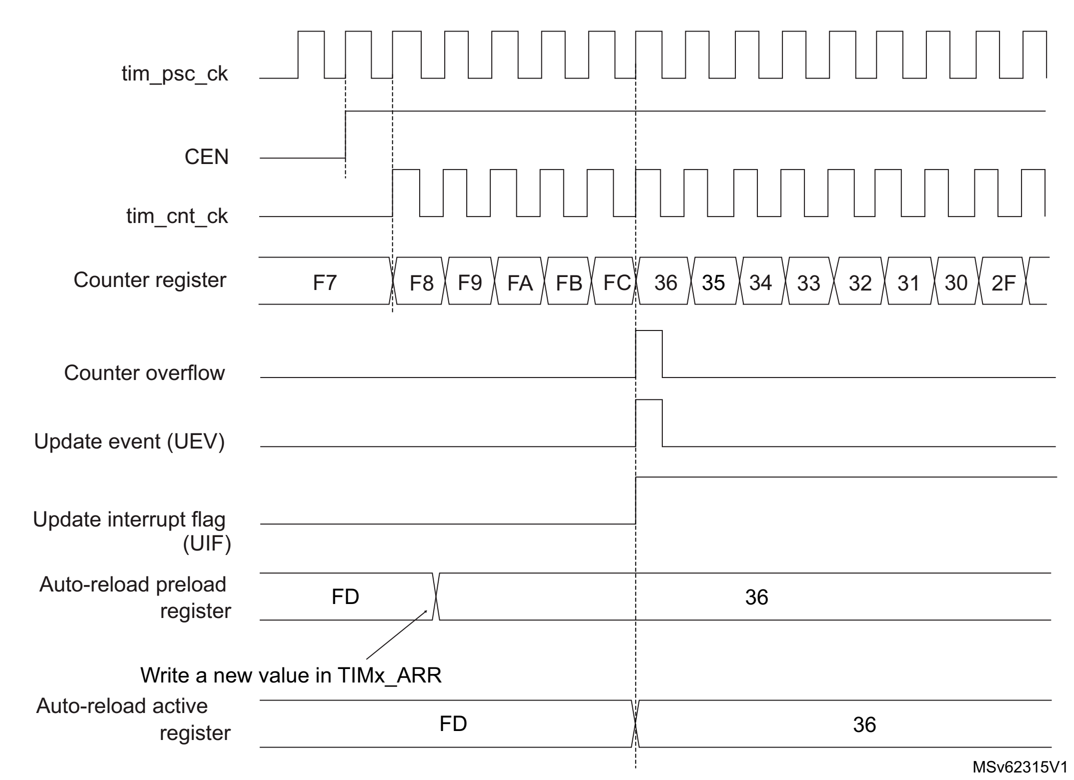


Figure 377. Counter timing diagram, Update event with ARPE=1 (counter overflow)



29.4.5 Clock selection

The counter clock can be provided by the following clock sources:

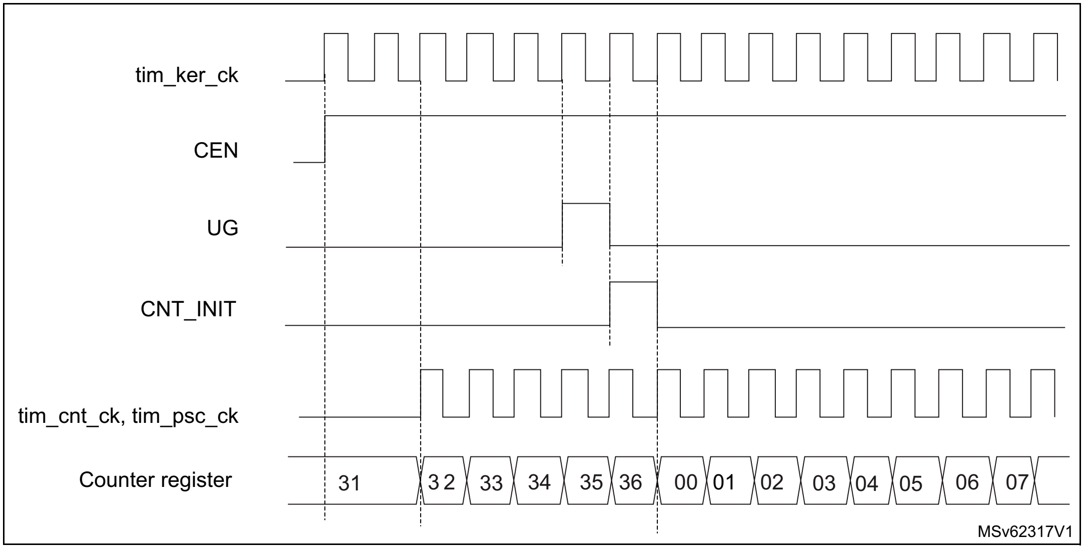
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, Timer 1 can be configured to act as a prescaler for Timer 2. Refer to *: Using one timer as prescaler for another timer on page 1300* for more details.

Internal clock source (tim_ker_ck)

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 tim_ker_ck.

Figure 378 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

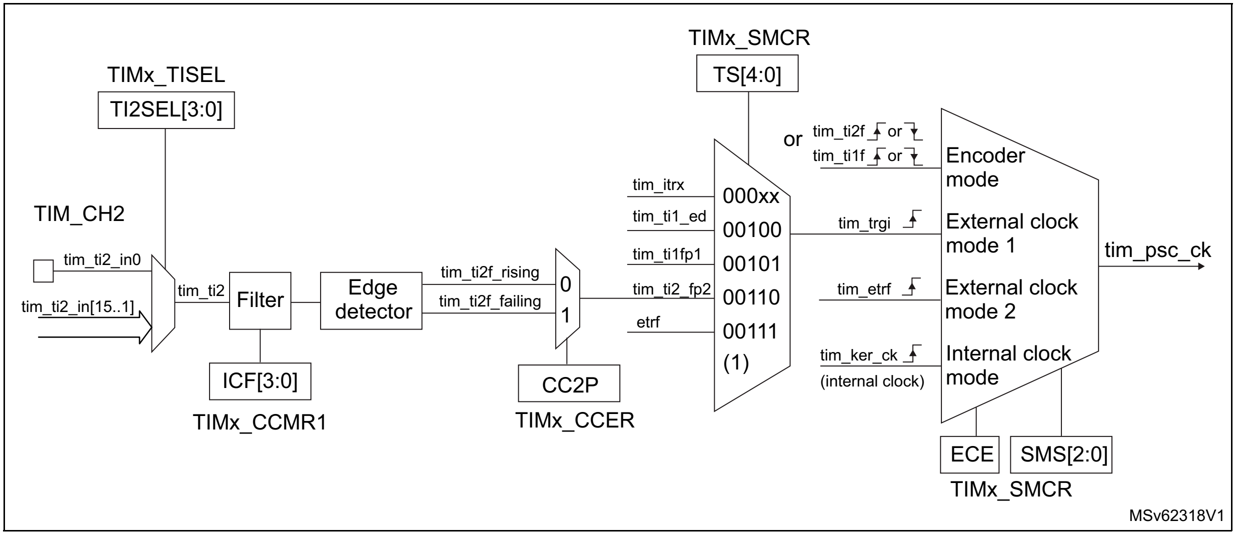
Figure 378. Control circuit in normal mode, internal clock divided by 1



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 379. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111: tim_itr[15..0].

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] 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 tim_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).

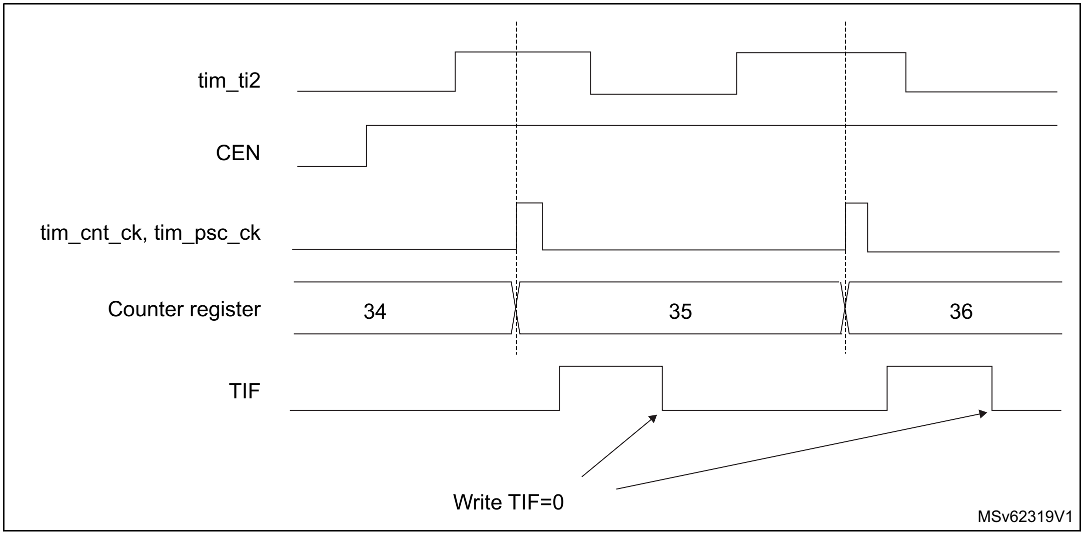
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

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 tim_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 tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 380. Control circuit in external clock mode 1



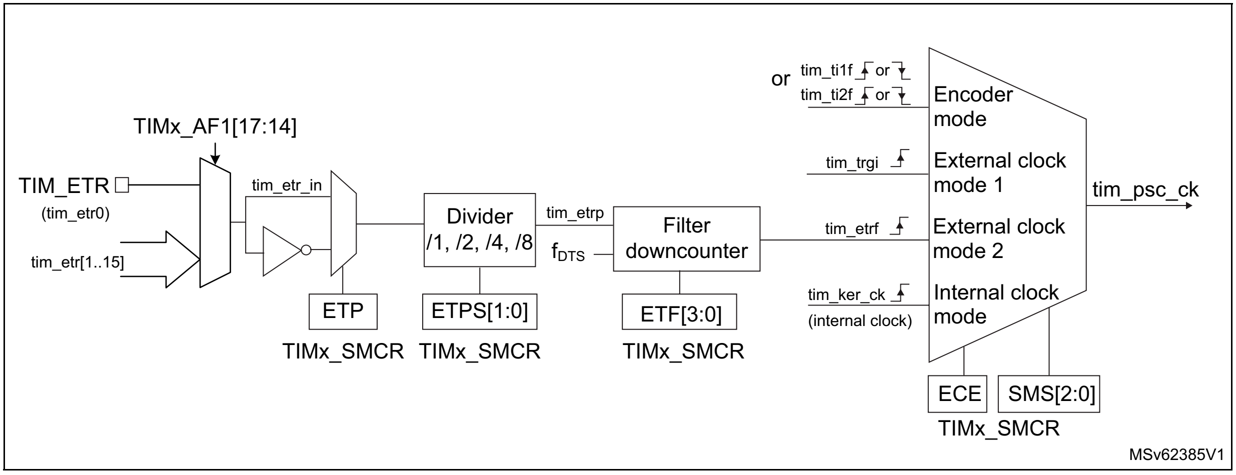
External clock source mode 2

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 tim_etr_in.

Figure 381 gives an overview of the external trigger input block.

Figure 381. External trigger input block



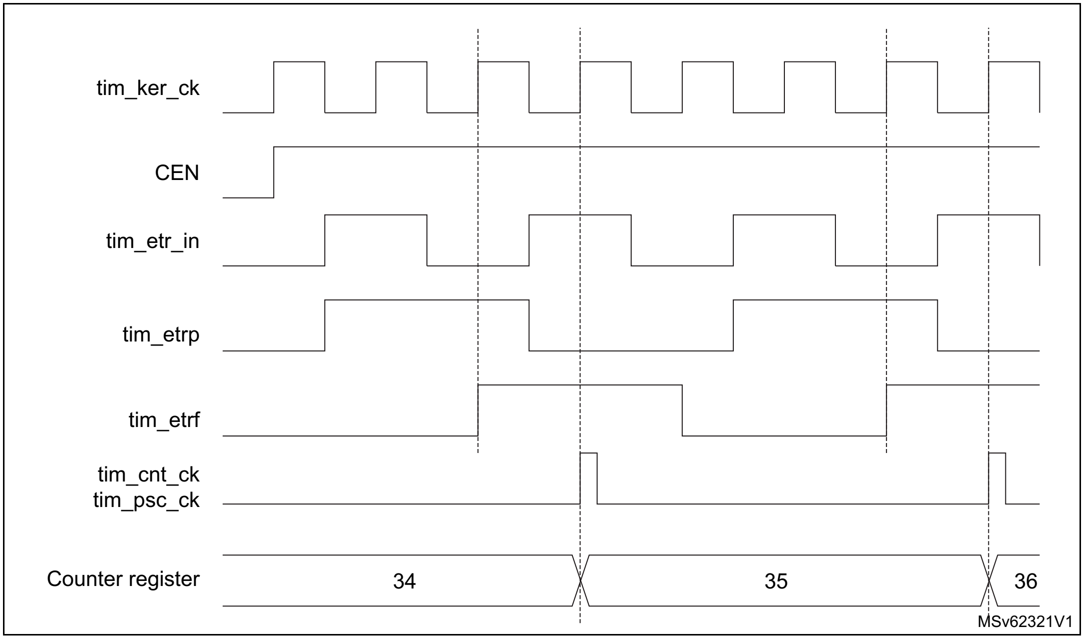
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 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 tim_etr_in 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 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that 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 382. Control circuit in external clock mode 2



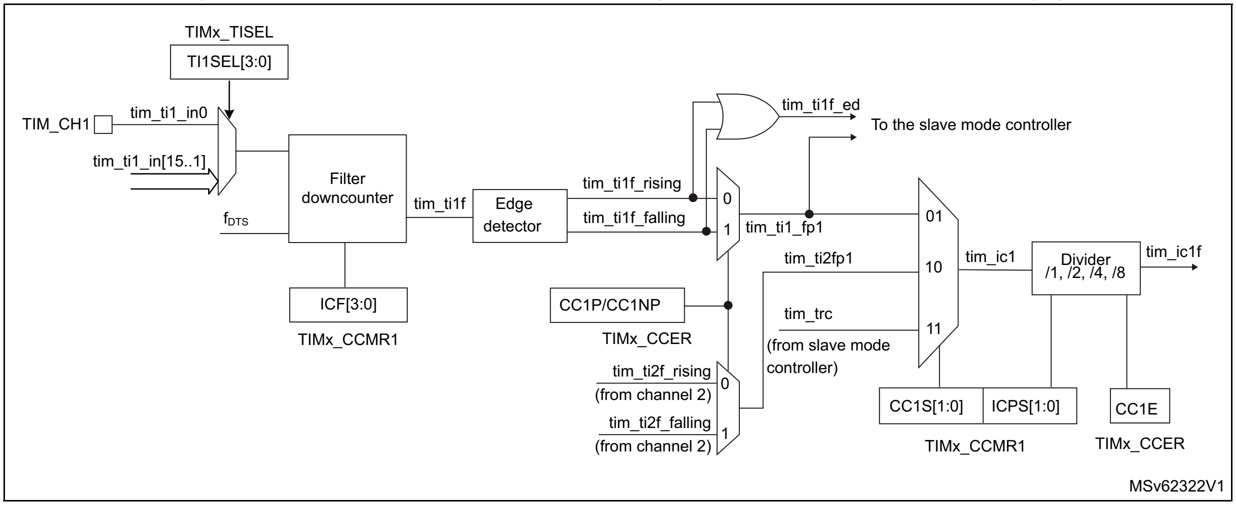
29.4.6 Capture/compare channels

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 tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) 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 383. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 384. Capture/compare channel 1 main circuit

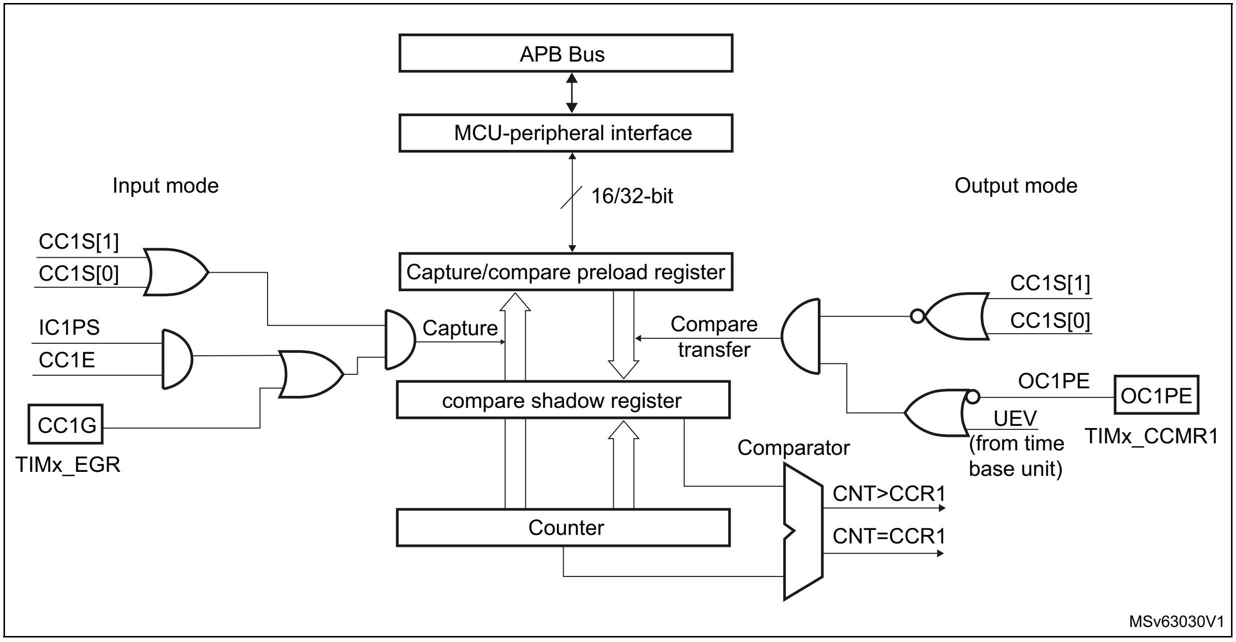
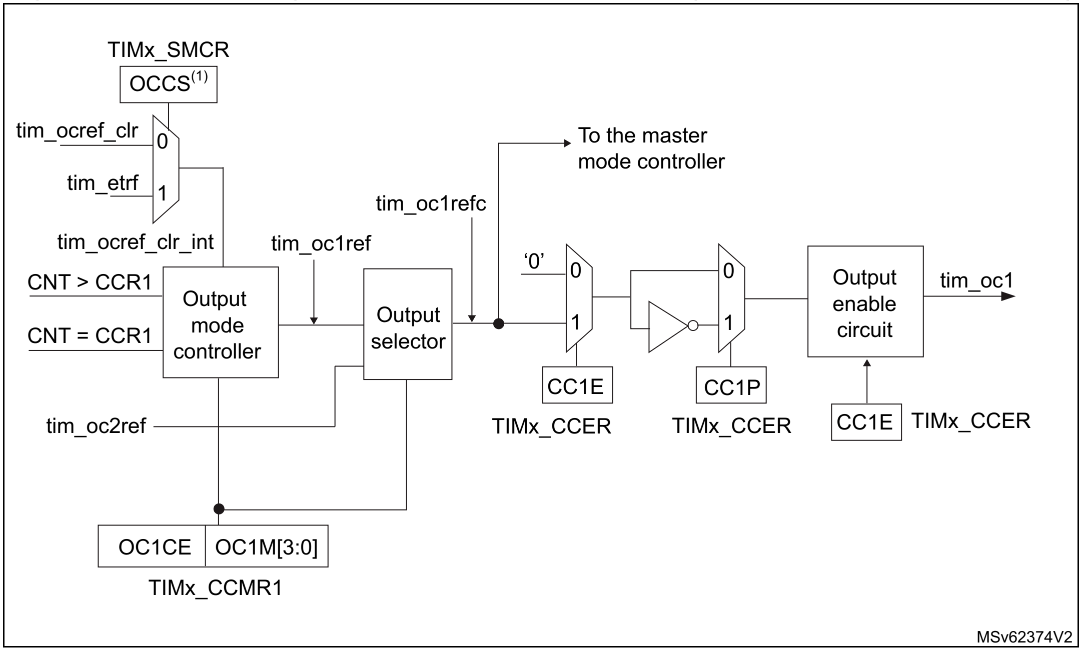


Figure 385. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

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.

29.4.7 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 it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15..0] 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 tim_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 needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_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 tim_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 tim_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.

29.4.8 PWM input mode

This mode allows to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges

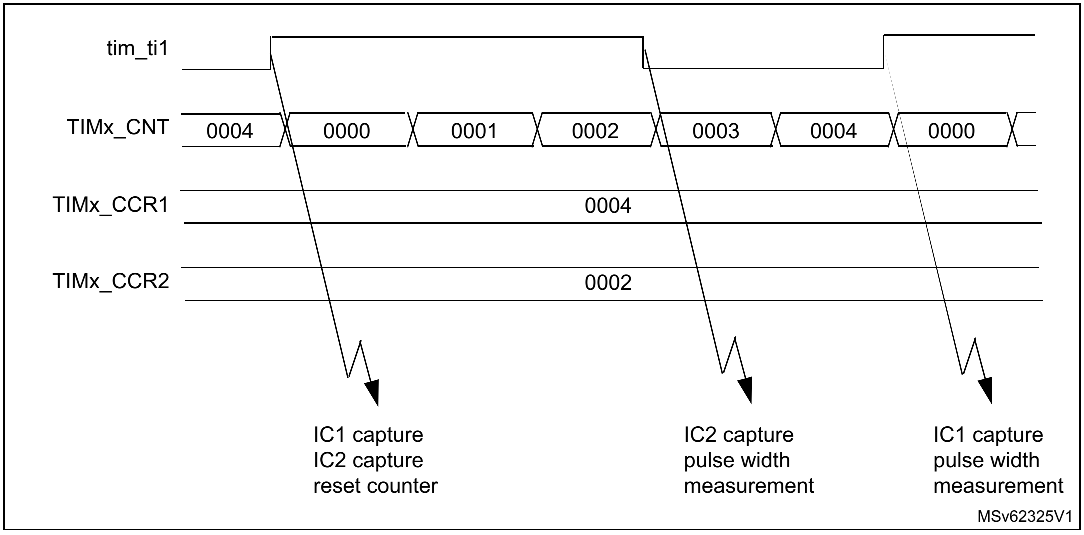
This mode is a particular case of input capture mode. The set-up procedure is similar wit hthe following differences:

- Two ICx signals are mapped on the same tim_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.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15..0] 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 (tim_ti1 selected).
3. Select the active polarity for tim_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 (tim_ti1 selected).
5. Select the active polarity for tim_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 (tim_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 386. 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 tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

29.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_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 (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_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.

29.4.10 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.

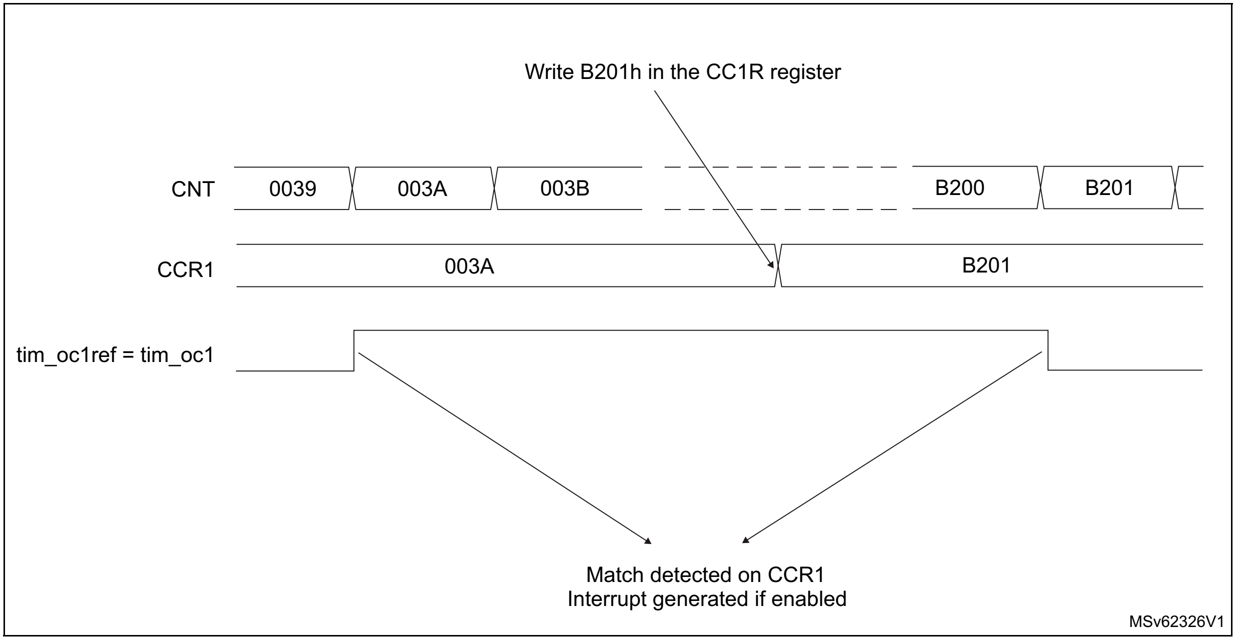
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   b) Write OCxPE = 0 to disable preload register
   c) Write CCxP = 0 to select active high polarity
   d) 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 387*.

Figure 387. Output compare mode, toggle on tim_oc1



29.4.11 PWM mode

Pulse width modulation mode allows 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 tim_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.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_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). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event
- 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.

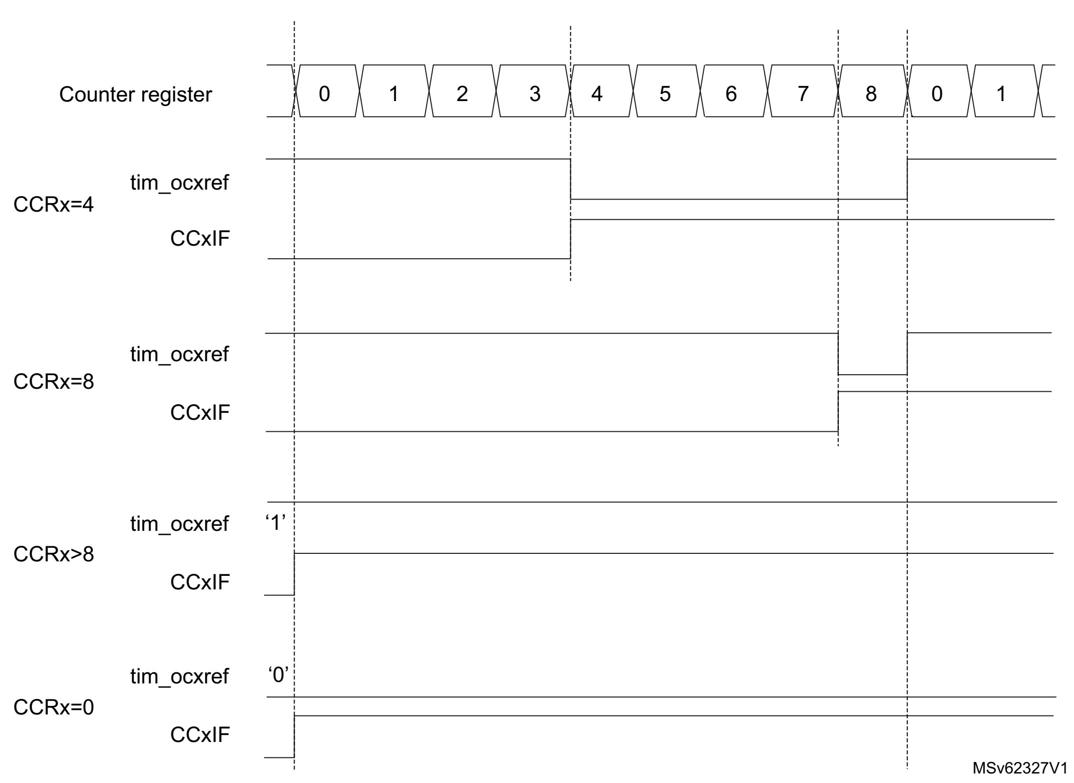
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 1238*.

In the following example, we consider PWM mode 1. The reference PWM signal tim_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 tim_ocxref is held at ‘1. If the compare value is 0 then tim_ocxref is held at ‘0. *Figure 388* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 388. 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 1242*.

In PWM mode 1, the reference signal tim_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 tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

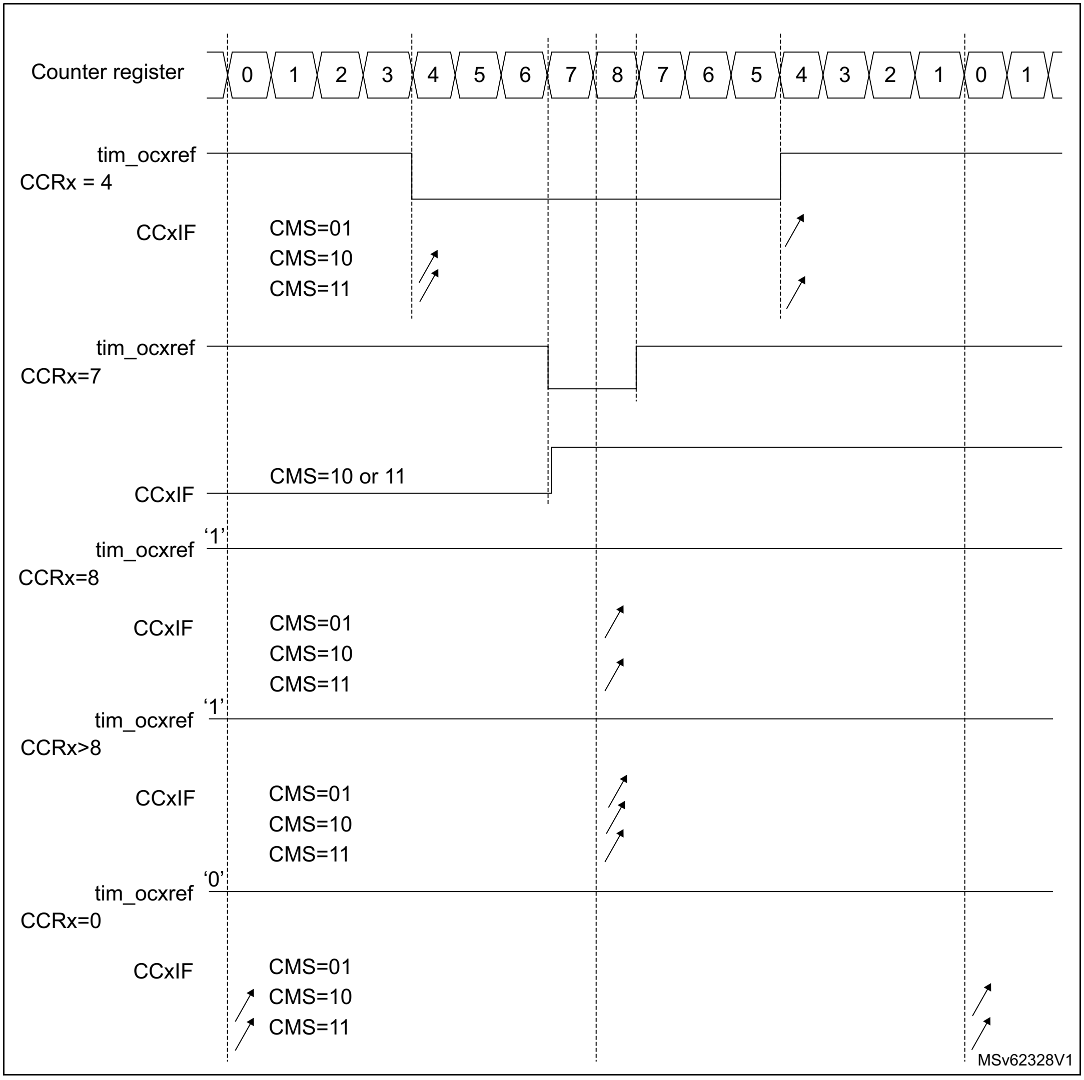
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending 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 1245*.

Figure 389 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 389. 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.

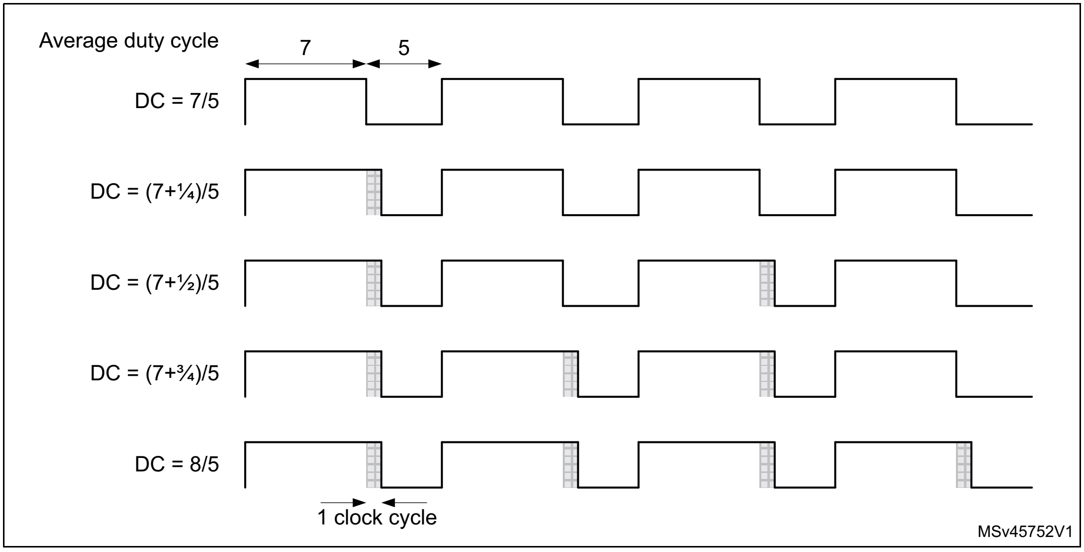
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting 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.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 390* below presents the dithering principle applied to 4 consecutive PWM cycles.

Figure 390. Dithering principle

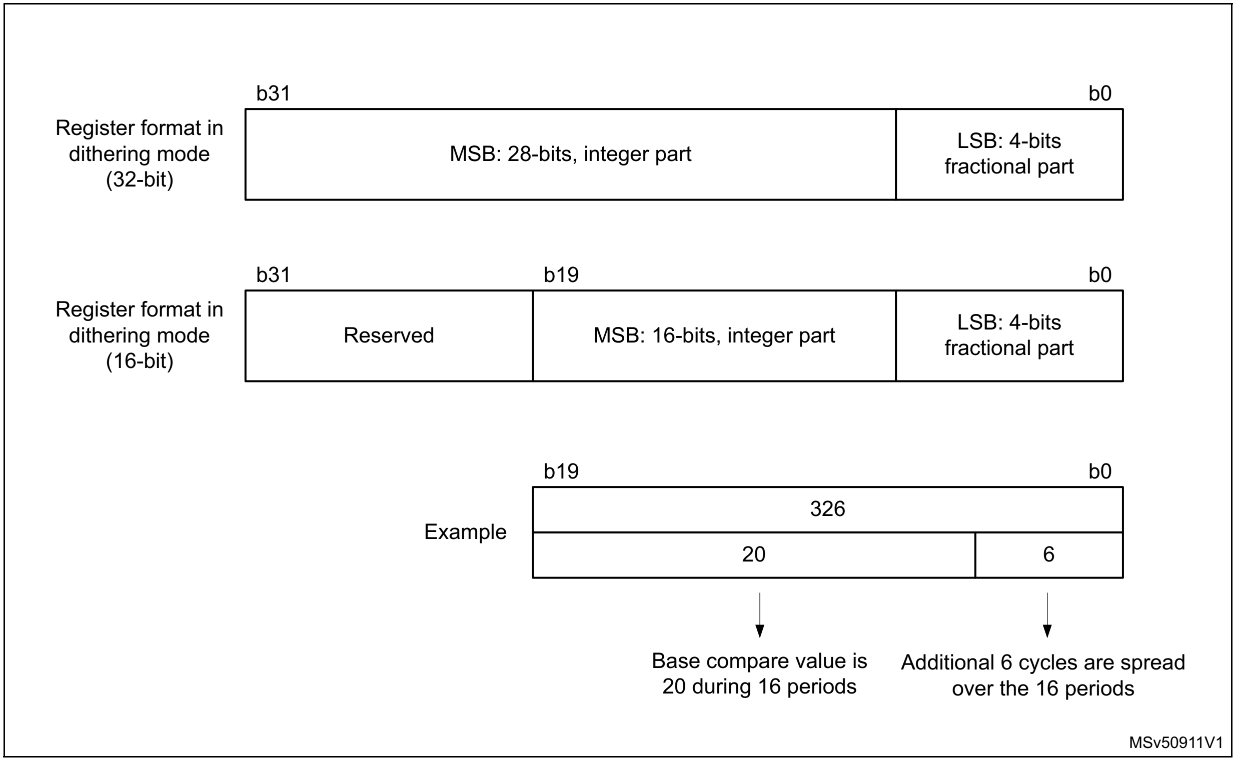


When the dithering mode is enabled, the register coding is changed as following (see *Figure 391* for example):

- The 4 LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by 4 places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: *The ARR and CCR values will be updated automatically if the DITHEN bit is set / reset (for instance, if ARR= 0x05 with DITHEN=0, it will be updated to ARR = 0x50 with DITHEN=1). The following sequence must be followed when resetting the DITHEN bit:*
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The DITHEN bit must be reset
4. The CCIF flags must be cleared
5. The CEN bit can be set (eventually with ARPE = 1)

Figure 391. Data format and register coding in dithering mode


Register format in dithering mode (32-bit)
b31
b0
MSB: 28-bits, integer part
LSB: 4-bits fractional part
Register format in dithering mode (16-bit)
b31
b19
b0
Reserved
MSB: 16-bits, integer part
LSB: 4-bits fractional part
Example
b19
b0
326
20
6
Base compare value is 20 during 16 periods
Additional 6 cycles are spread over the 16 periods
MSv50911V1

The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on the *Figure 392* and *Figure 393* below, the dithering mode allows to increase the PWM resolution.

Figure 392. PWM resolution vs frequency (16-bit mode)

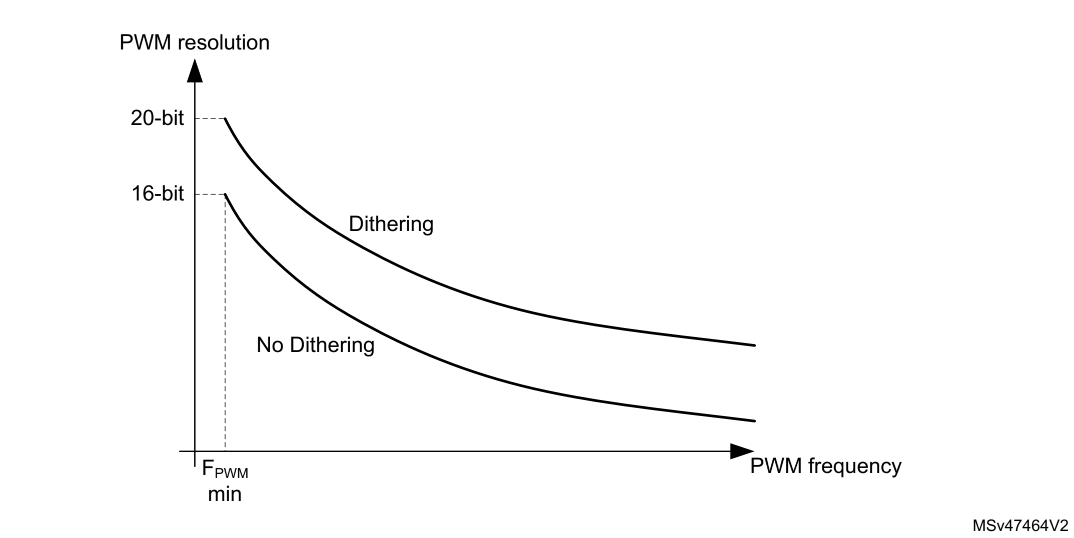
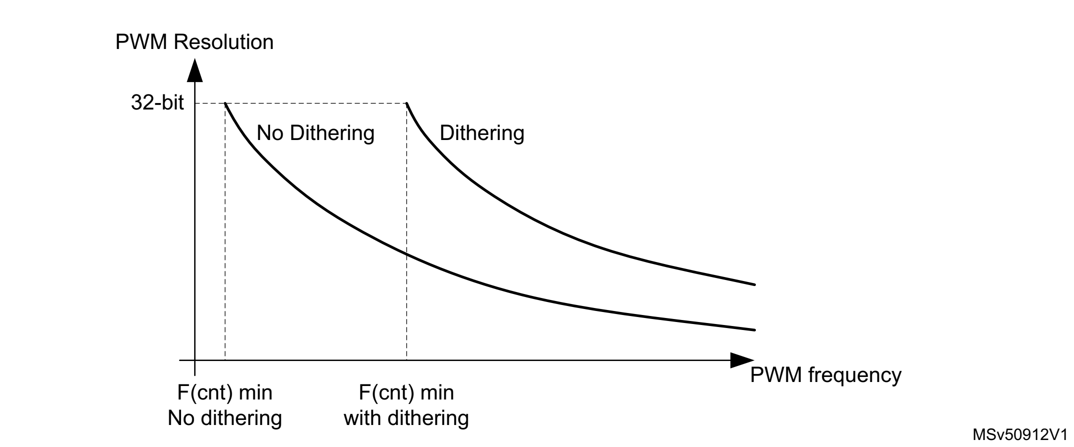
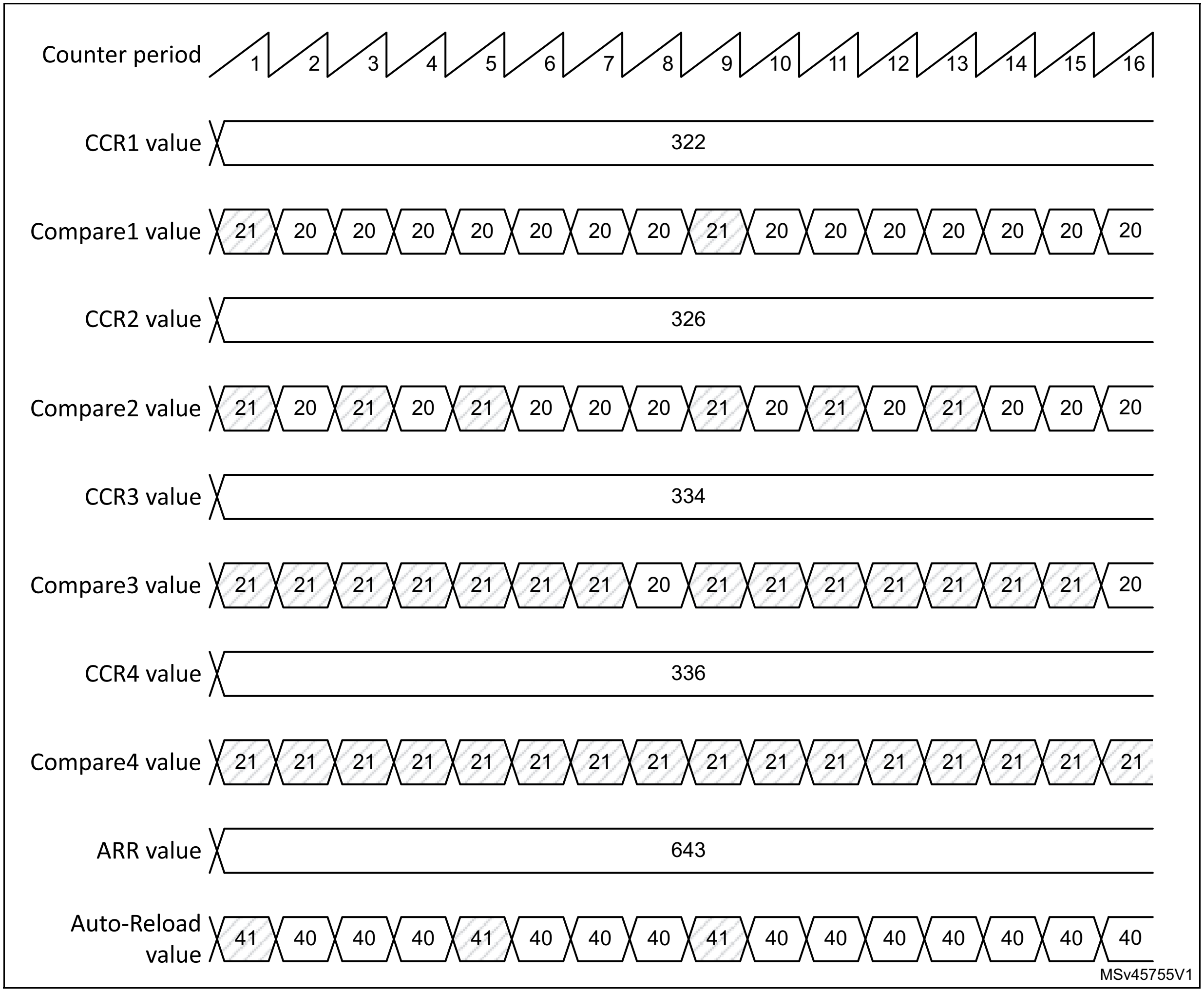


Figure 393. PWM resolution vs frequency (32-bit mode)



The duty cycle and / or period changes are spread over 16 consecutive periods, as described in the *Figure 394* below.

Figure 394. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 277* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 277. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to '00'). In this case, the dithering pattern is applied over 8 consecutive PWM periods, considering the up and down counting phases as shown in the *Figure 395* below.

Figure 395. Dithering effect on duty cycle in center-aligned PWM mode

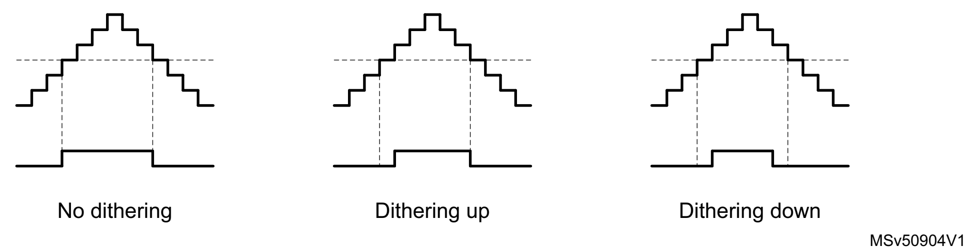


Table 278 below shows how the dithering pattern is added in center-aligned PWM mode.

Table 278. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

29.4.12 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 registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

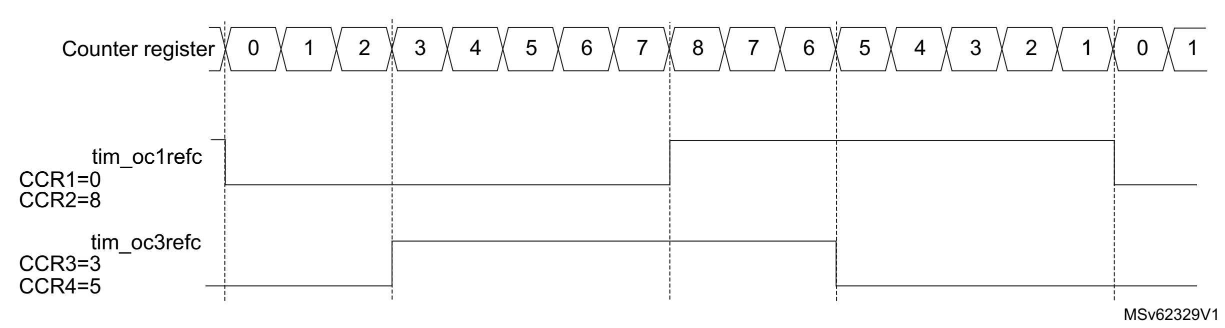
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

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 tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Figure 396 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 2).

Figure 396. Generation of 2 phase-shifted PWM signals with 50% duty cycle



29.4.13 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, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

– tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2

– tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_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.

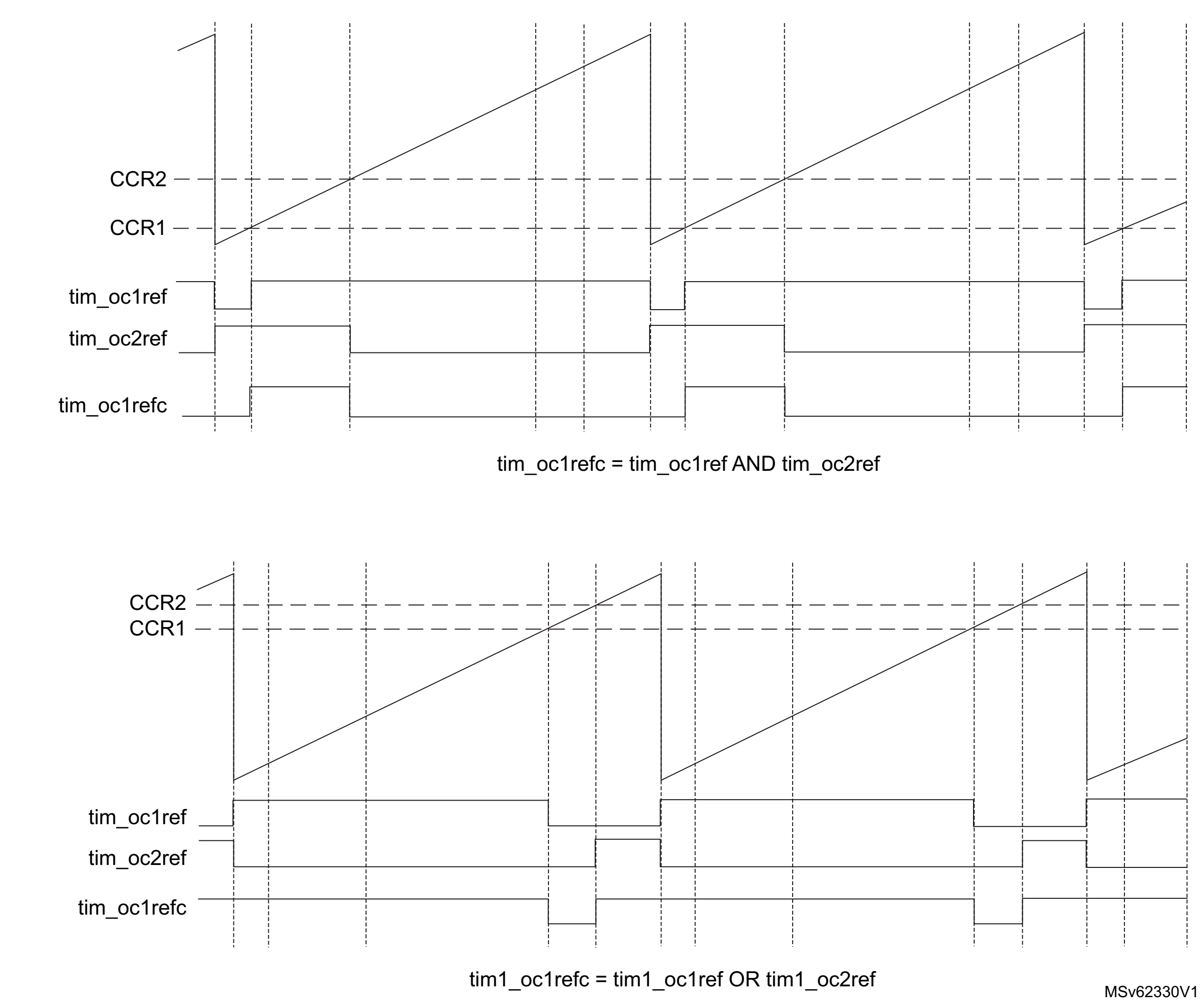
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 397 shows an example of signals that can be generated using combined 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 397. Combined PWM mode on channels 1 and 3



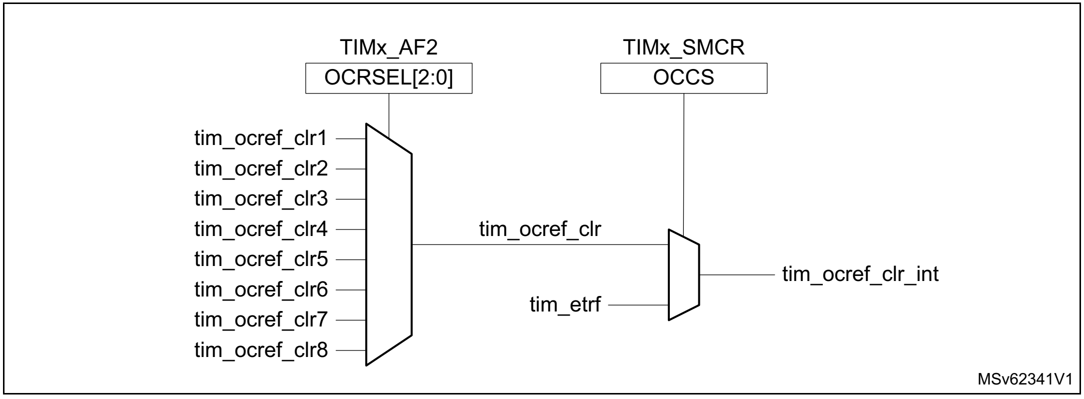
29.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_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.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 29.3: TIM2/TIM3/TIM4/TIM5 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7..0] inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown in *Figure 398* below.

Figure 398. OCREF_CLR input selection multiplexer



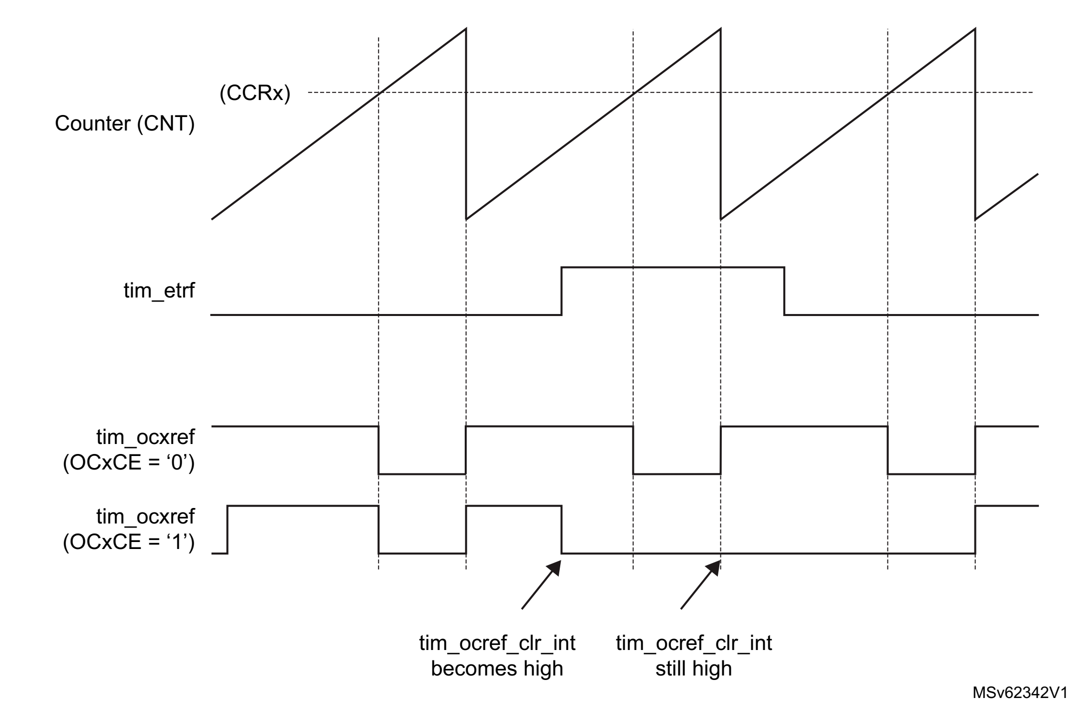
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in 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 399 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 399. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

29.4.15 One-pulse mode

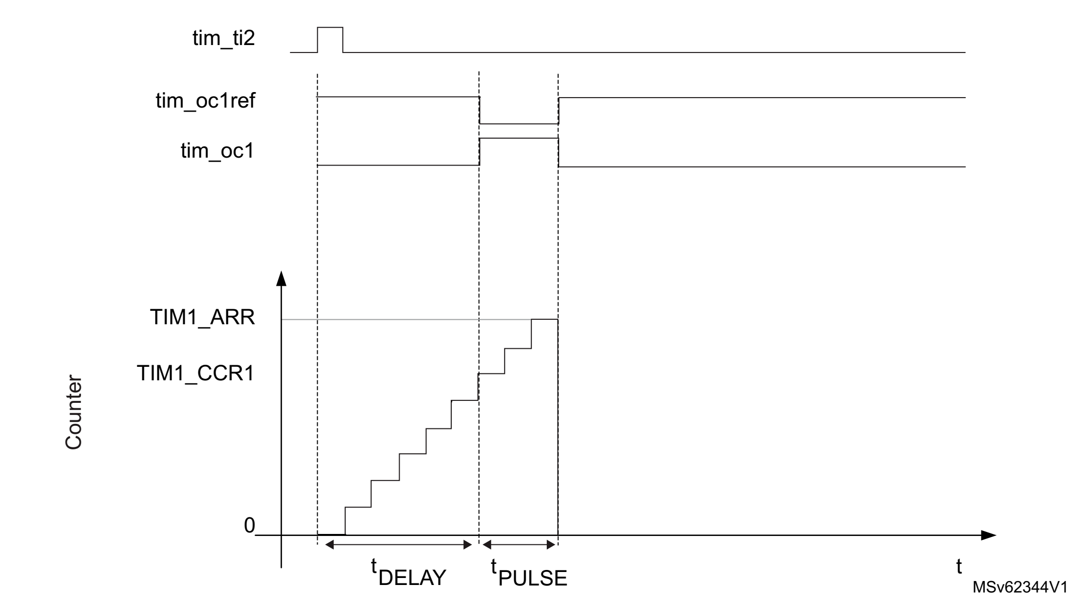
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 400. Example of one-pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let’s use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15..0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P=0 and CC2NP=’0’ in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS=00110 in the TIMx_SMCR register.
5. tim_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 tim_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: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_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 tim_ocxref (and tim_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.

29.4.16 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 29.4.15*:

- 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.

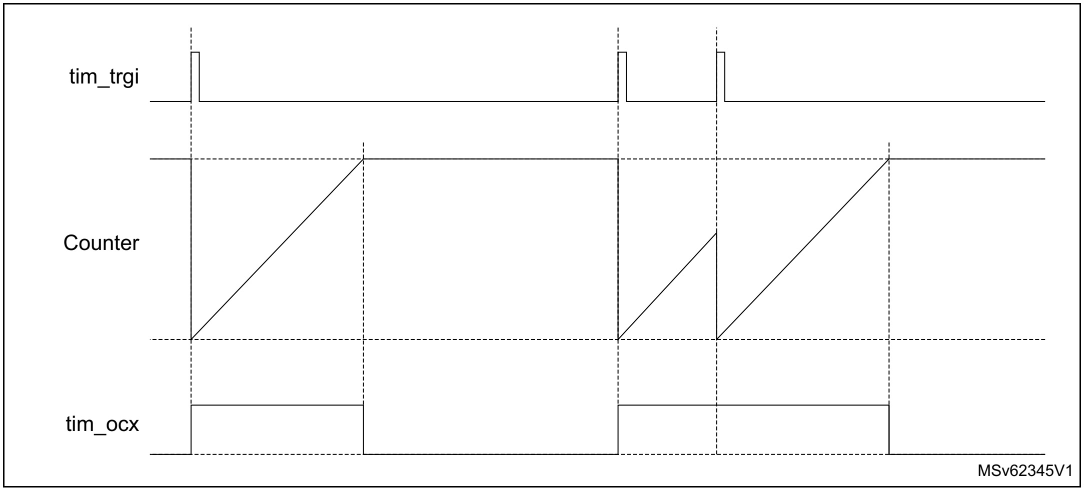
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

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 401. Retriggerable one pulse mode

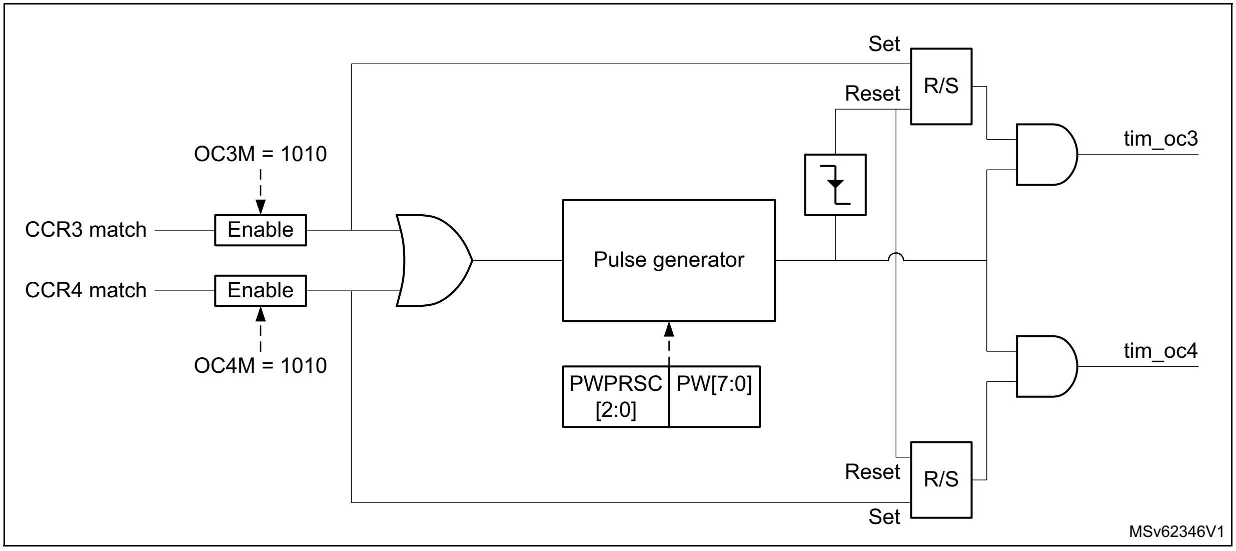


29.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

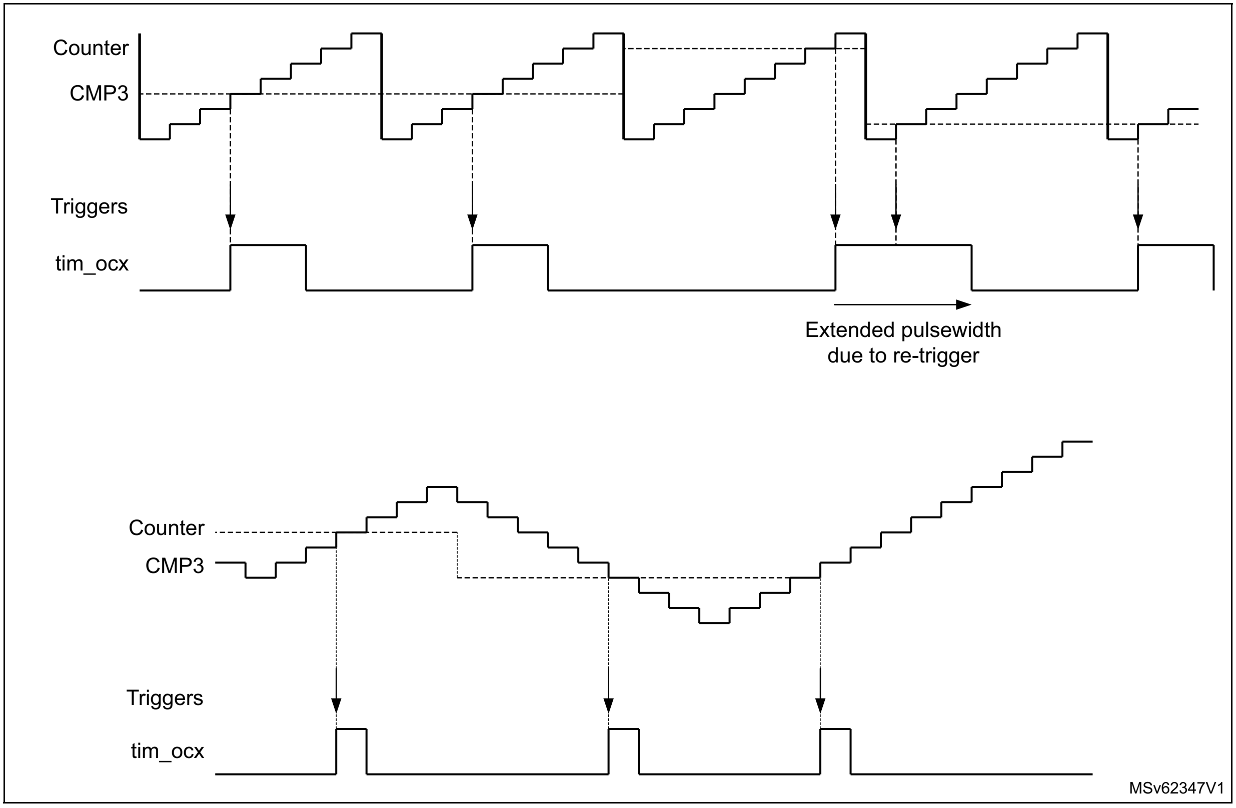
This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on the *Figure 402* below.

Figure 402. Pulse generator circuitry



The *Figure 403* below shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 403. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bit fields in TIMx_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

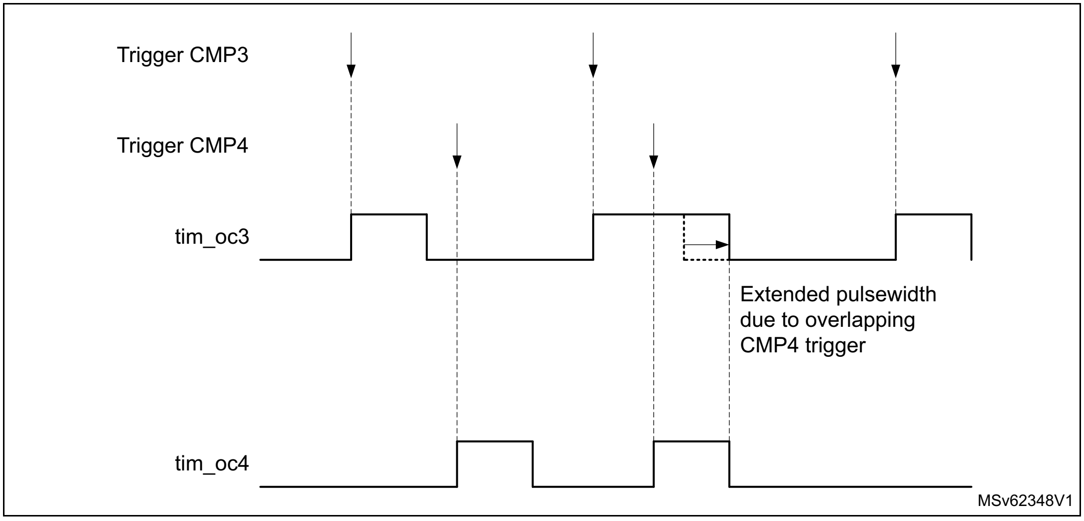
where $t_{PWG} = (2^{(PWPRSC[2:0])})$ x $t_{tim\_ker\_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is on-going, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the 1st arriving trigger is extended (because of the re-trigger), while the pulse width of the last arriving trigger is correct (as shown on the Figure 404 below).

Figure 404. Extended pulse width in case of concurrent triggers



29.4.18 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS=‘0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS=0010 if it is counting on tim_ti2 edges only and SMS=0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_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.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 279*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1=tim_ti1 if not filtered and not inverted, tim_ti2fp2=tim_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 (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_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. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

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 tim_ti1 and tim_ti2 do not switch at the same time.

Table 279. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	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 the external trigger input and trigger a counter reset.

Figure 405 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, tim_ti1fp1 mapped on tim_ti1)
- CC2S= 01 (TIMx_CCMR2 register, tim_ti2fp2 mapped on tim_ti2)
- CC1P and CC1NP = ‘0’ (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1=tim_ti1)
- CC2P and CC2NP = ‘0’ (TIMx_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2=tim_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 405. Example of counter operation in encoder interface mode

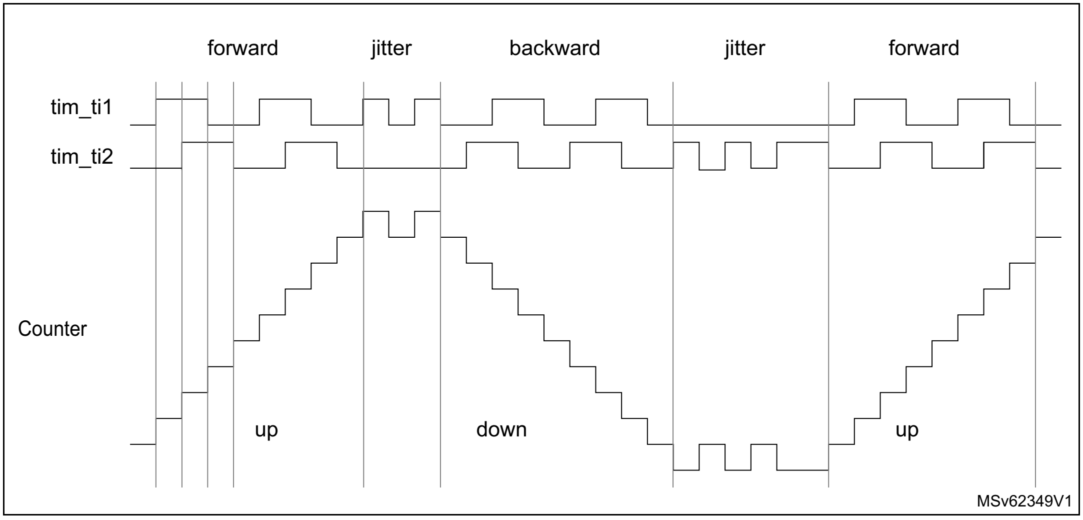
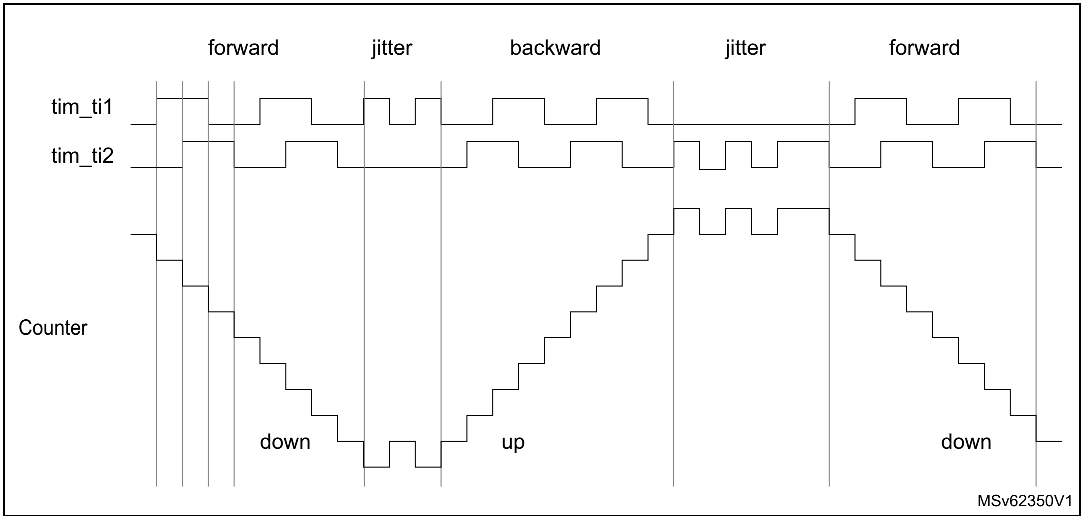


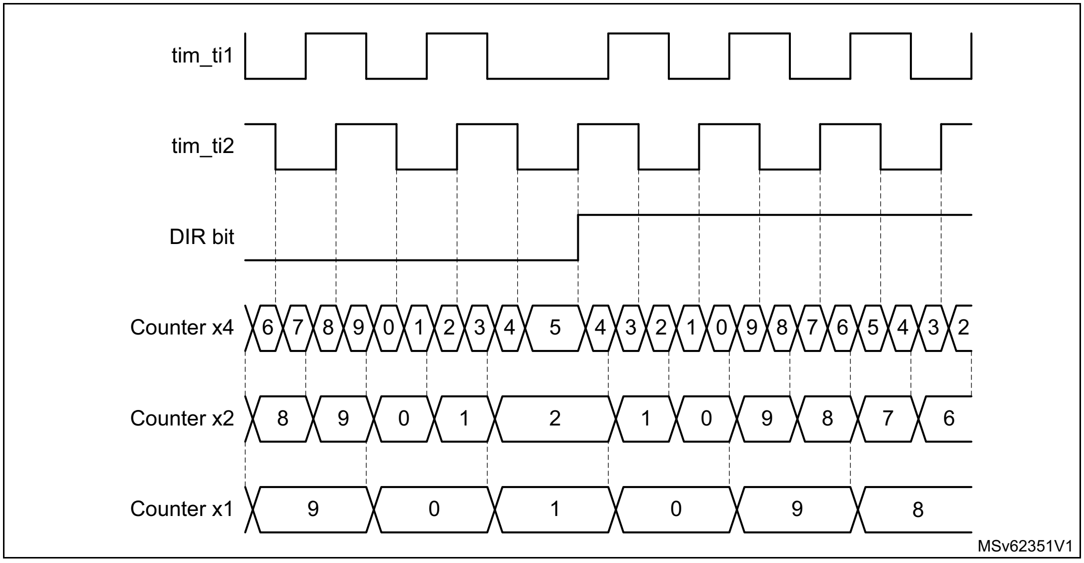
Figure 406 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P=1).

Figure 406. Example of encoder interface mode with tim_ti1fp1 polarity inverted



The *Figure 407* below shows the timer counter value during a speed reversal, for various counting modes.

Figure 407. Quadrature encoder counting modes



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).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support other types of encoders.

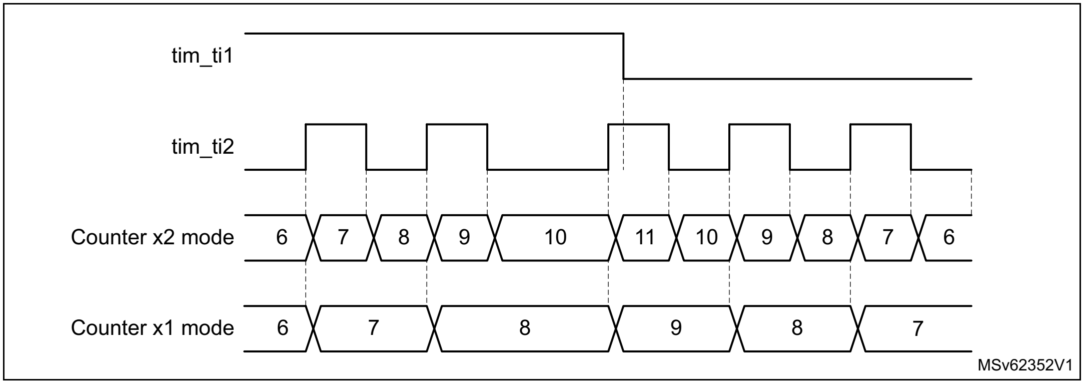
In the "clock plus direction" mode shown on *Figure 408*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 408. Direction plus clock encoder mode



Directional Clock encoder mode

In the “directional clock” mode on *Figure 409*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 409* below) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 410* below).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 409* below), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 410* below).

Figure 409. Directional clock encoder mode (CC1P = CC2P = 0)

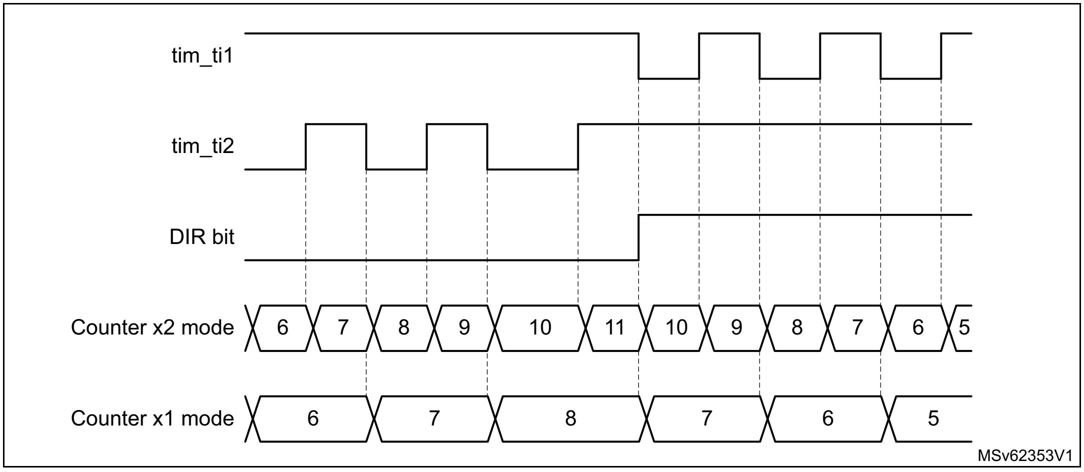
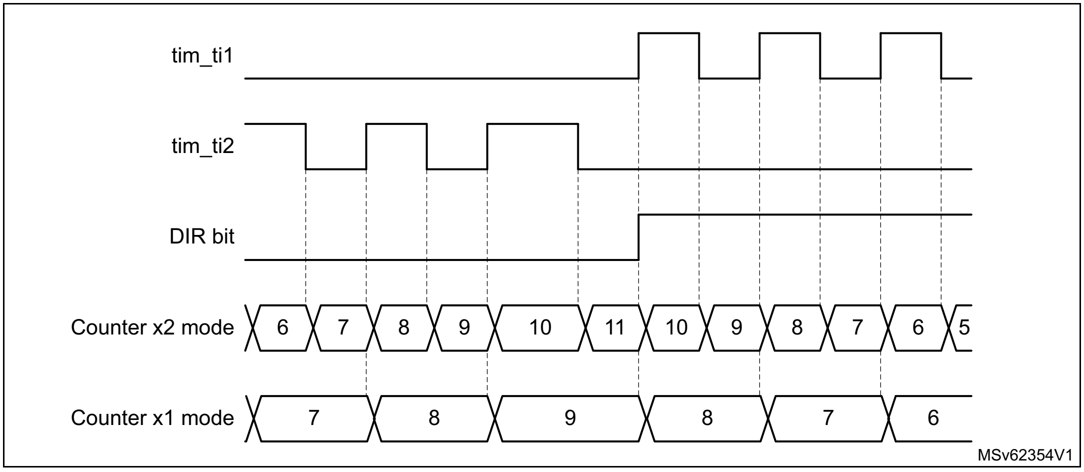


Figure 410. Directional clock encoder mode (CC1P = CC2P = 1)



The *Table 280* here-below details how the directional clock mode operates, for any input transition.

Table 280. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP=0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP=1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP=0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP=1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index Input

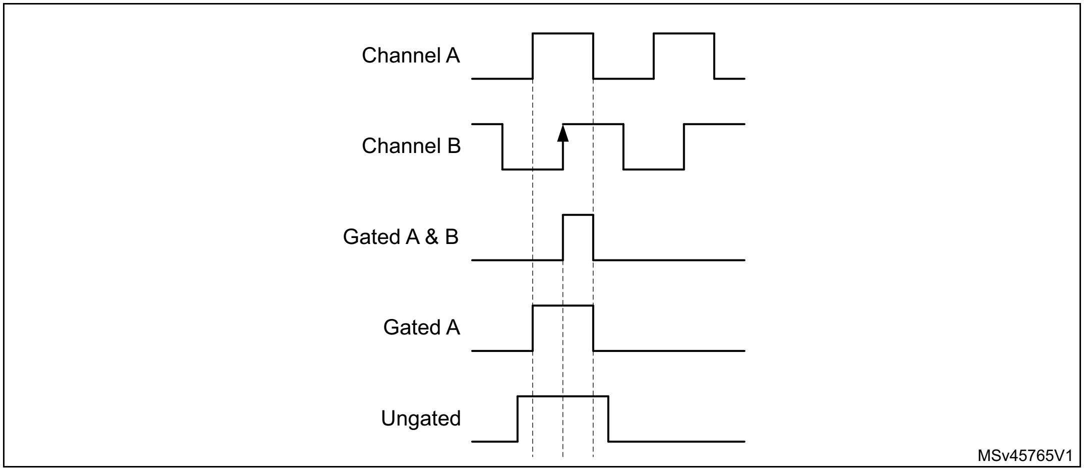
The counter can be reset by an Index signal coming from the encoder, indicating an absolute reference position. The Index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Commercially available encoders are proposed with several options for index pulse conditioning, as per the *Figure 411* below:

- gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulse width is up to one channel period, without any alignment to the edges

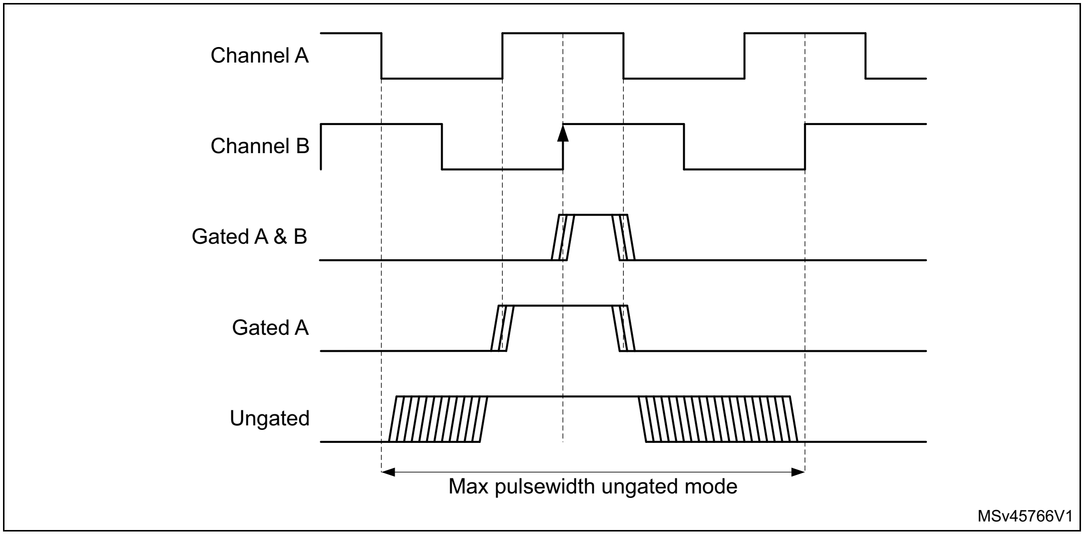
Figure 411. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 412* below.

In ungated mode, the signal must be strictly below 2 encoder periods. If the pulse width is greater or equal to 2 encoder period, the counter is reset multiple times.

Figure 412. Jittered Index signals



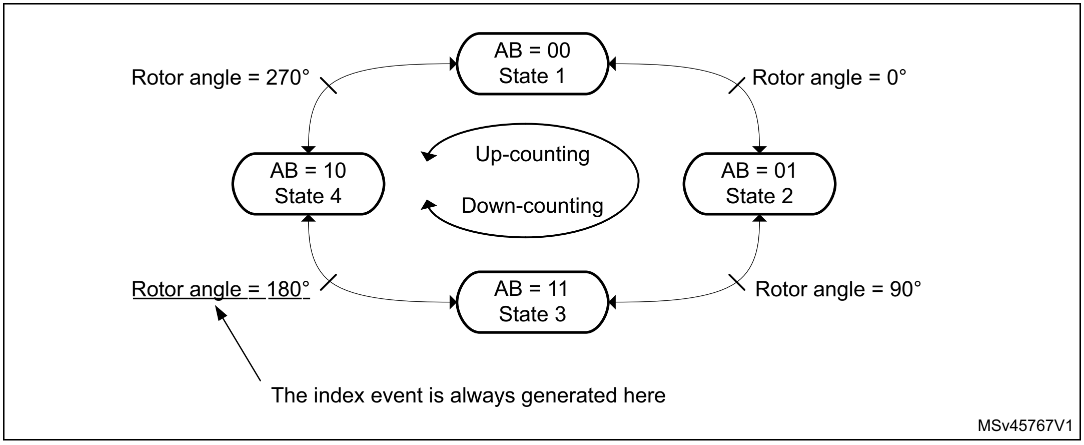
The timer supports the 3 gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (i.e. channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The Index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0)
- The counter is set to TIMx_ARR when down counting

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. The *Figure 413* below shows at which position is the index generated, for a simplistic example (an encoder providing 4 edges par mechanical rotation).

Figure 413. Index generation for IPOS[1:0] = 11



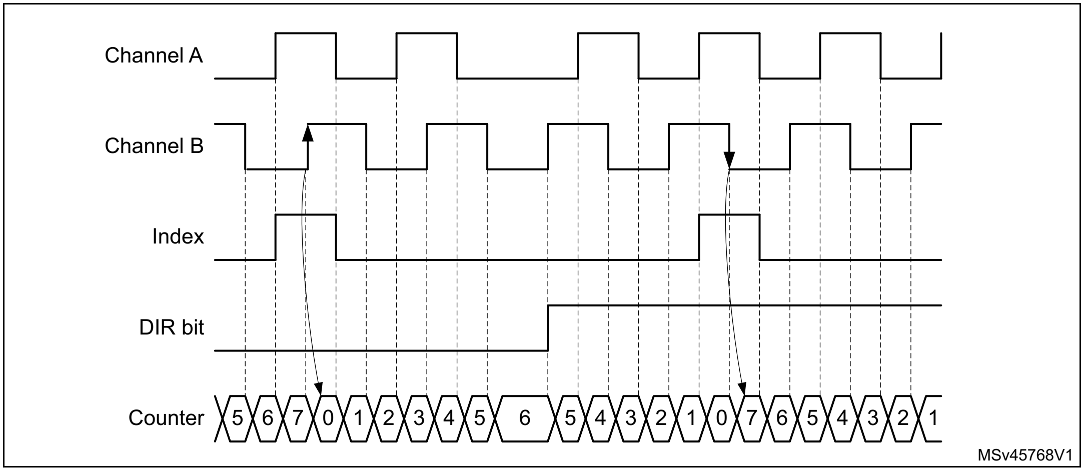
The *Figure 414* below presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is '11' (ChA=1, ChB=1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the '11' state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

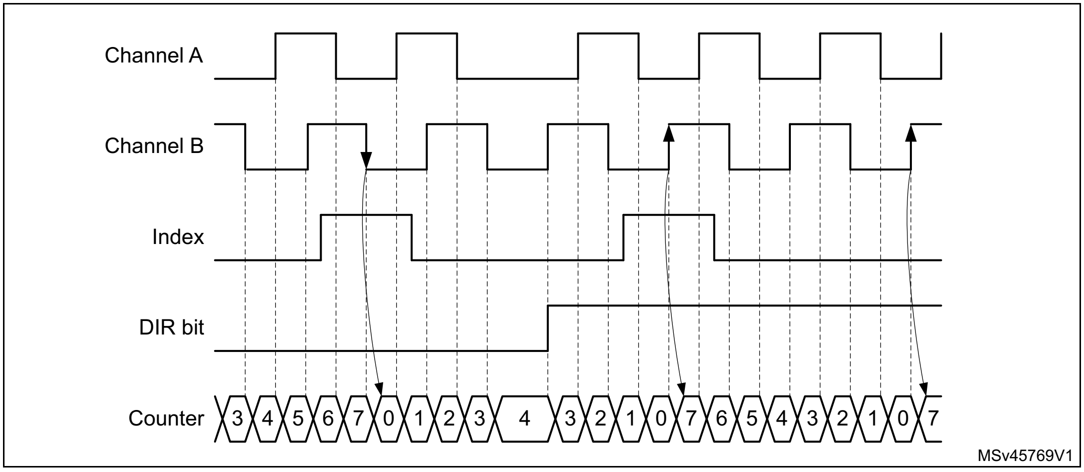
The arrows are indicating on which transition is the index event interrupt generated.

Figure 414. Counter reading with index gated on channel A (IPOS[1:0] = 11)



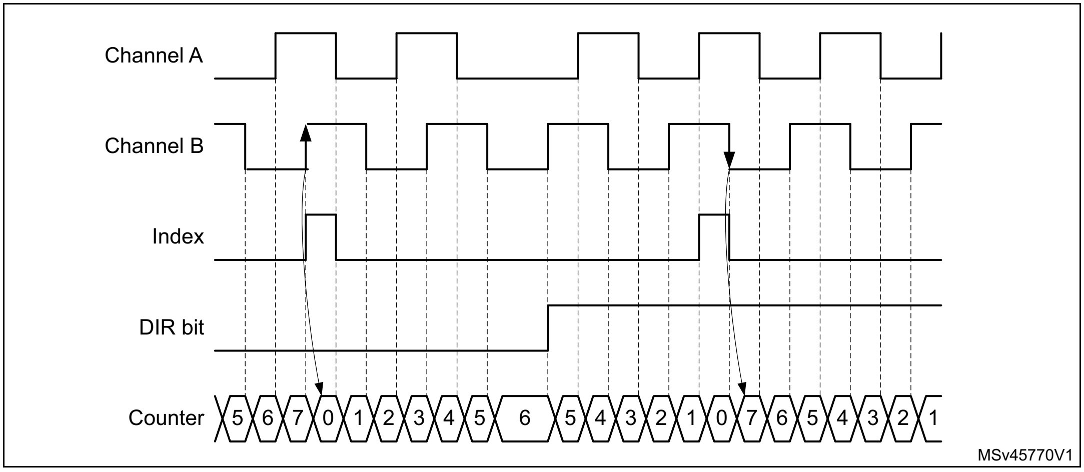
The *Figure 415* below presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 415. Counter reading with index ungated (IPOS[1:0] = 00)



The *Figure 416* below shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 416. Counter reading with index gated on channel A and B



The *Figure 417* and *Figure 418* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 417. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

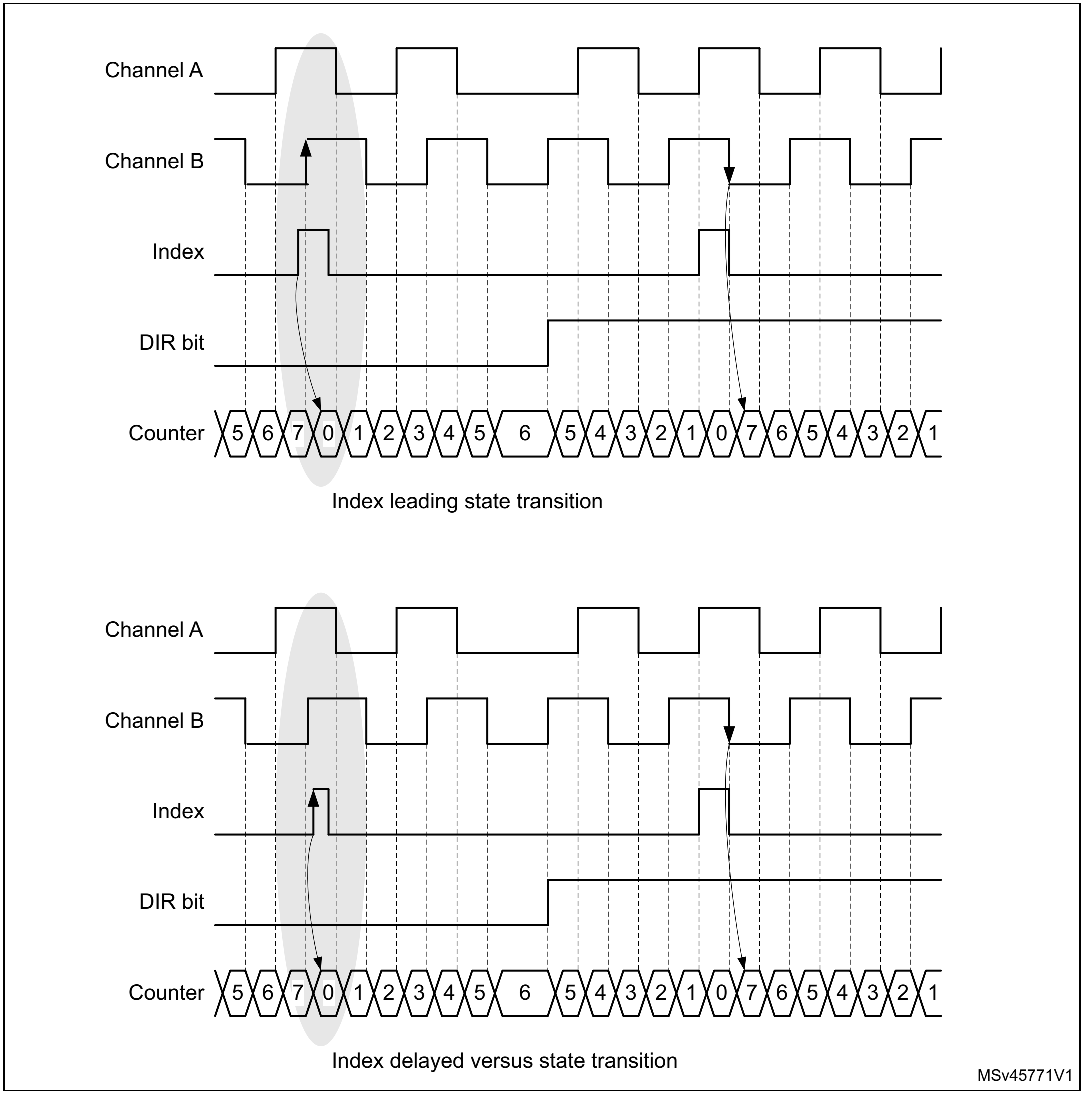
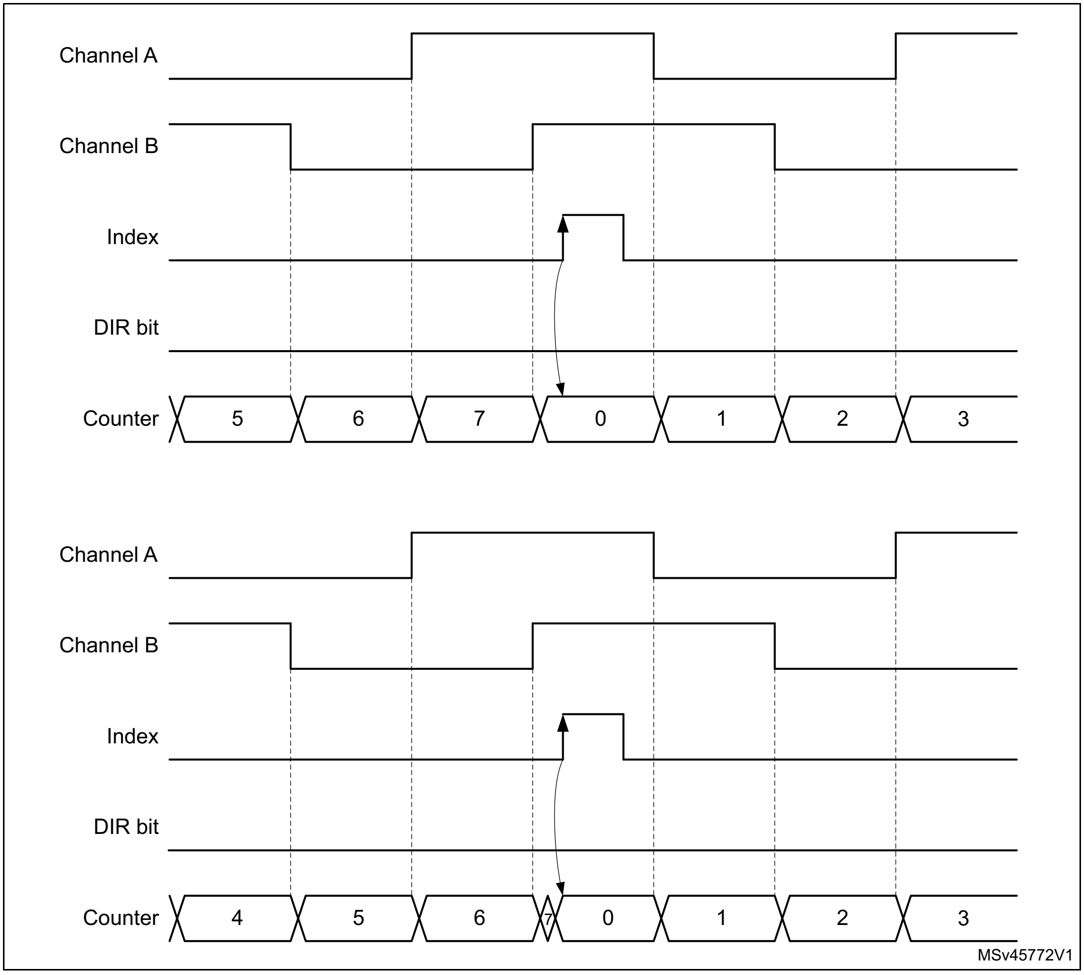
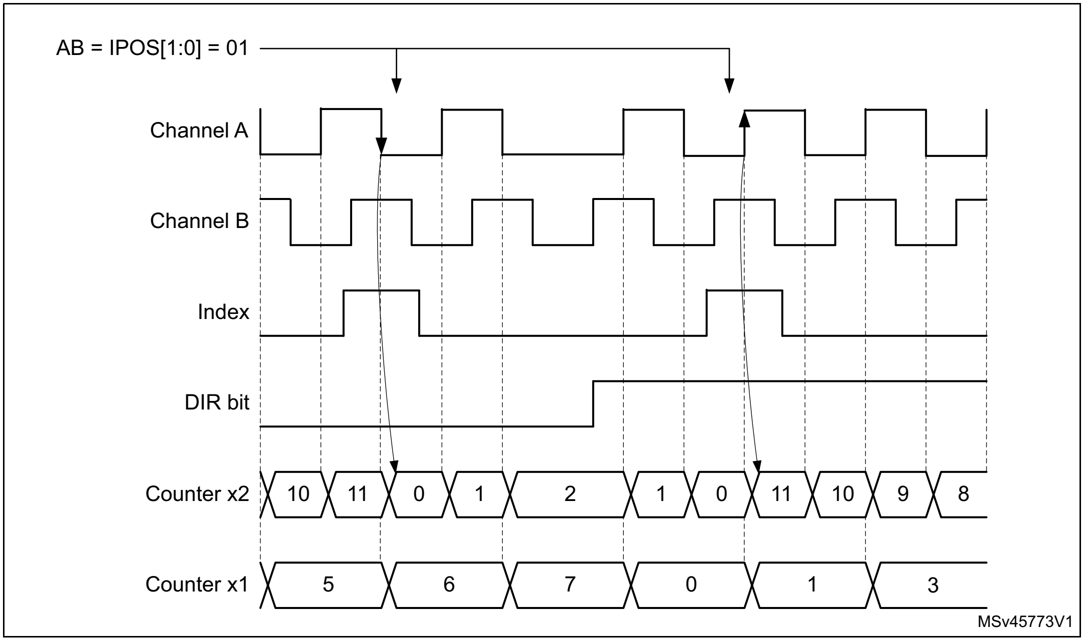


Figure 418. Counter reset Narrow index pulse (closer view, ARR = 0x07)



The *Figure 419* below shows how the index is managed in x1 and x2 modes.

Figure 419. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)



Directional index sensitivity

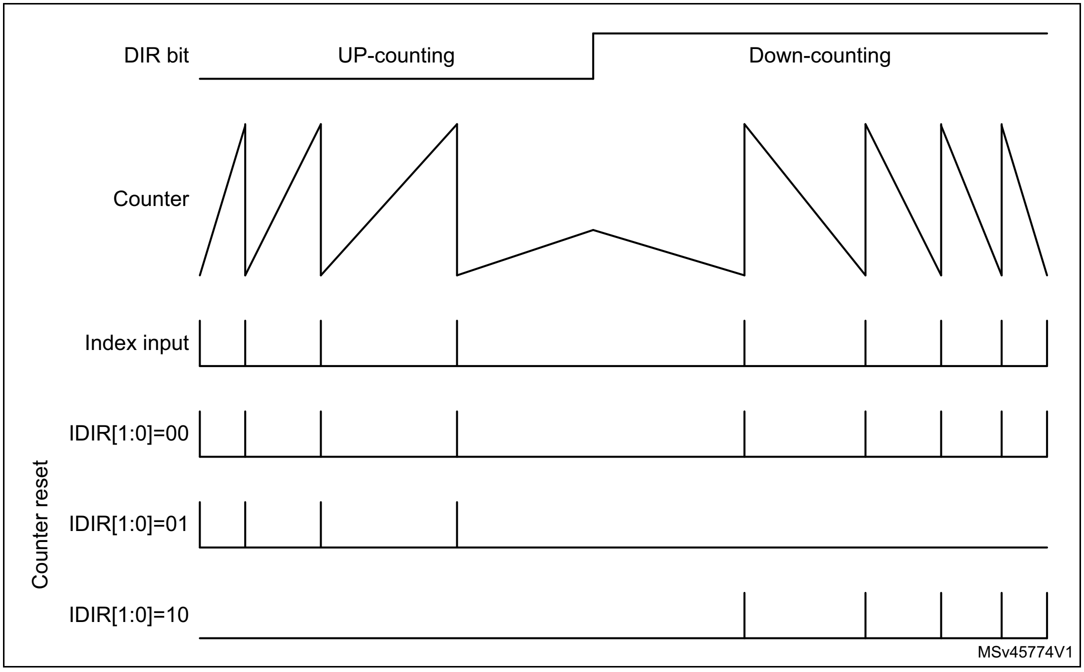
The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

The *Figure 420* below shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

Note: The IDR[1:0] bitfield must be written when IE bit is reset (index mode disabled).

Note: The directional index sensitivity is not supported in clock + direction mode. When SMS[3:0] = 1010 or 1011, the IDIR[1:0] must be set to 00.

Figure 420. Directional index sensitivity

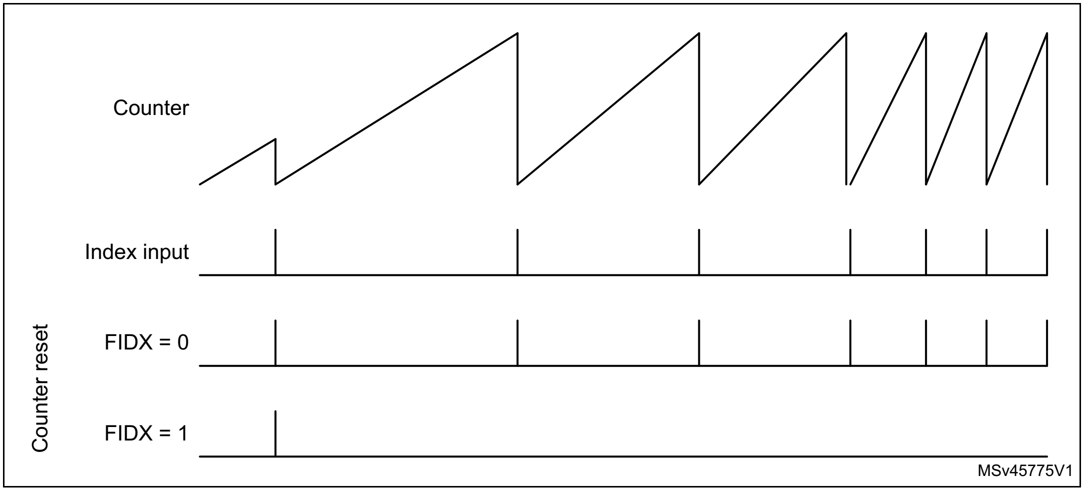


Special first index event management

The FIDX bit in the TIMx_ECR register allows the Index to be taken only once, as shown on the *Figure 421* below. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be re-armed by writing the FIDX bit to 0 and setting it again to 1.

Note: When FIDX = 1, the index can be issued twice (IDXF flag set) if the direction changes at position 0 (index active).

Figure 421. Counter reset as function of FIDX bit setting



Index management in non-quadrature mode

The *Figure 422* and *Figure 423* below detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 422. Index behavior in clock + direction mode, IPOS[0] = 1

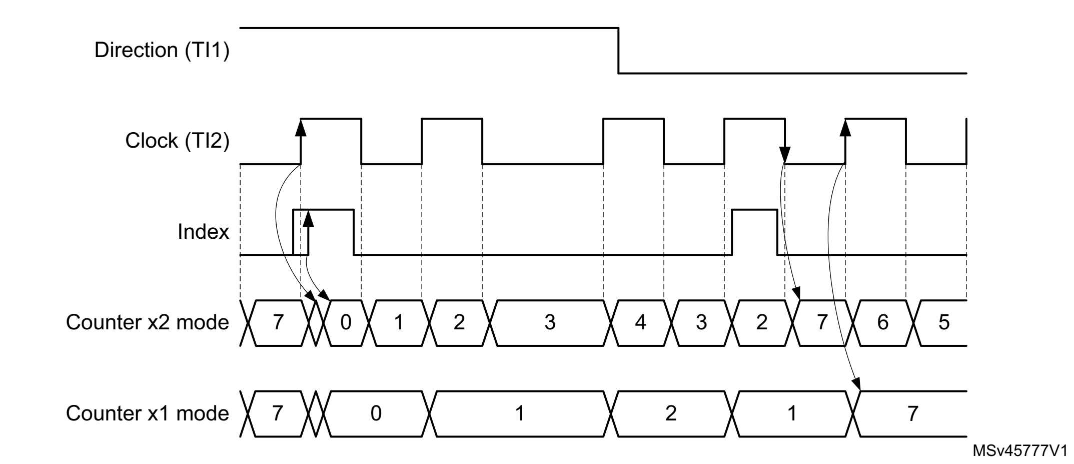
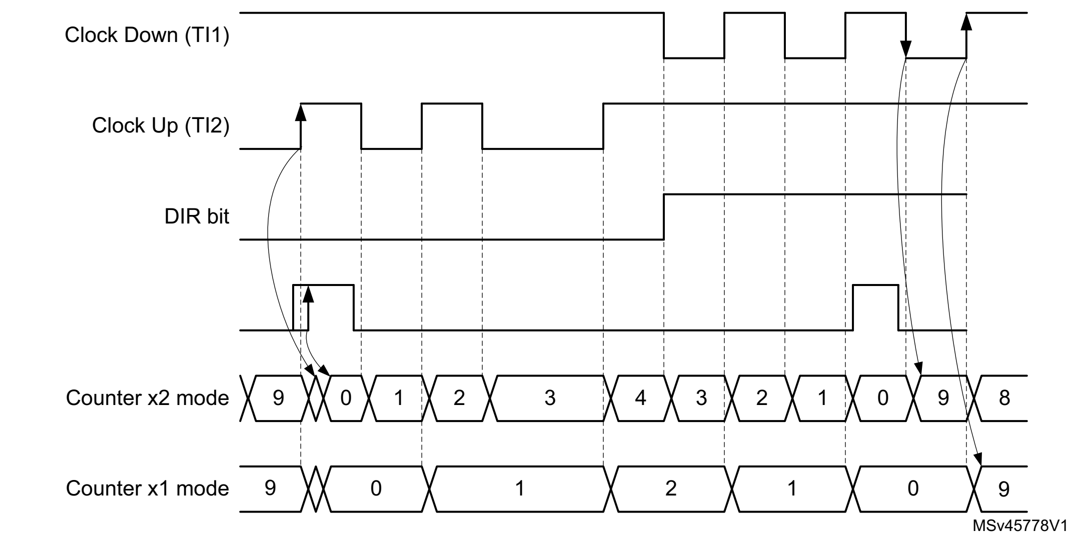


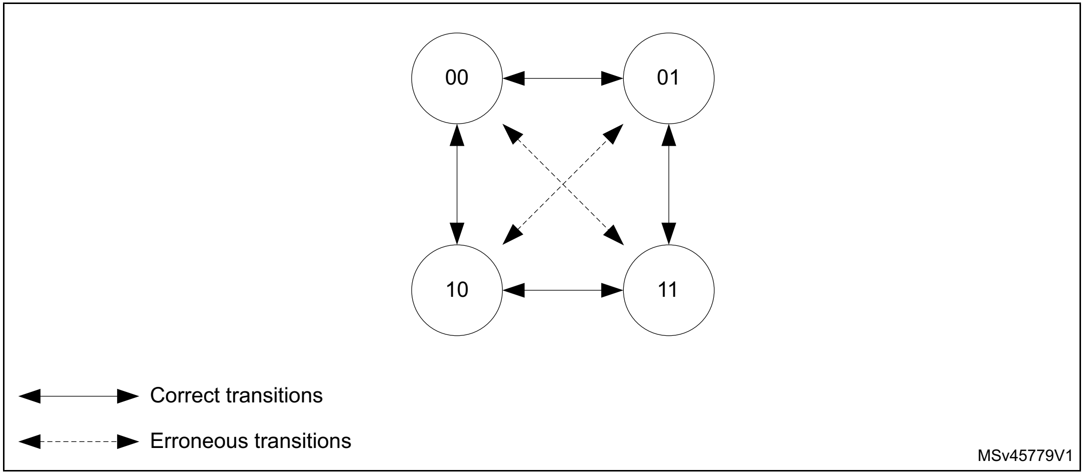
Figure 423. Index behavior in directional clock mode, IPOS[0] = 1



Encoder error management

For encoder configurations where 2 quadrature signals are available, it is possible to detect transition errors. The reading on the 2 inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on the *Figure 424.* below. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 424. State diagram for quadrature encoded signals



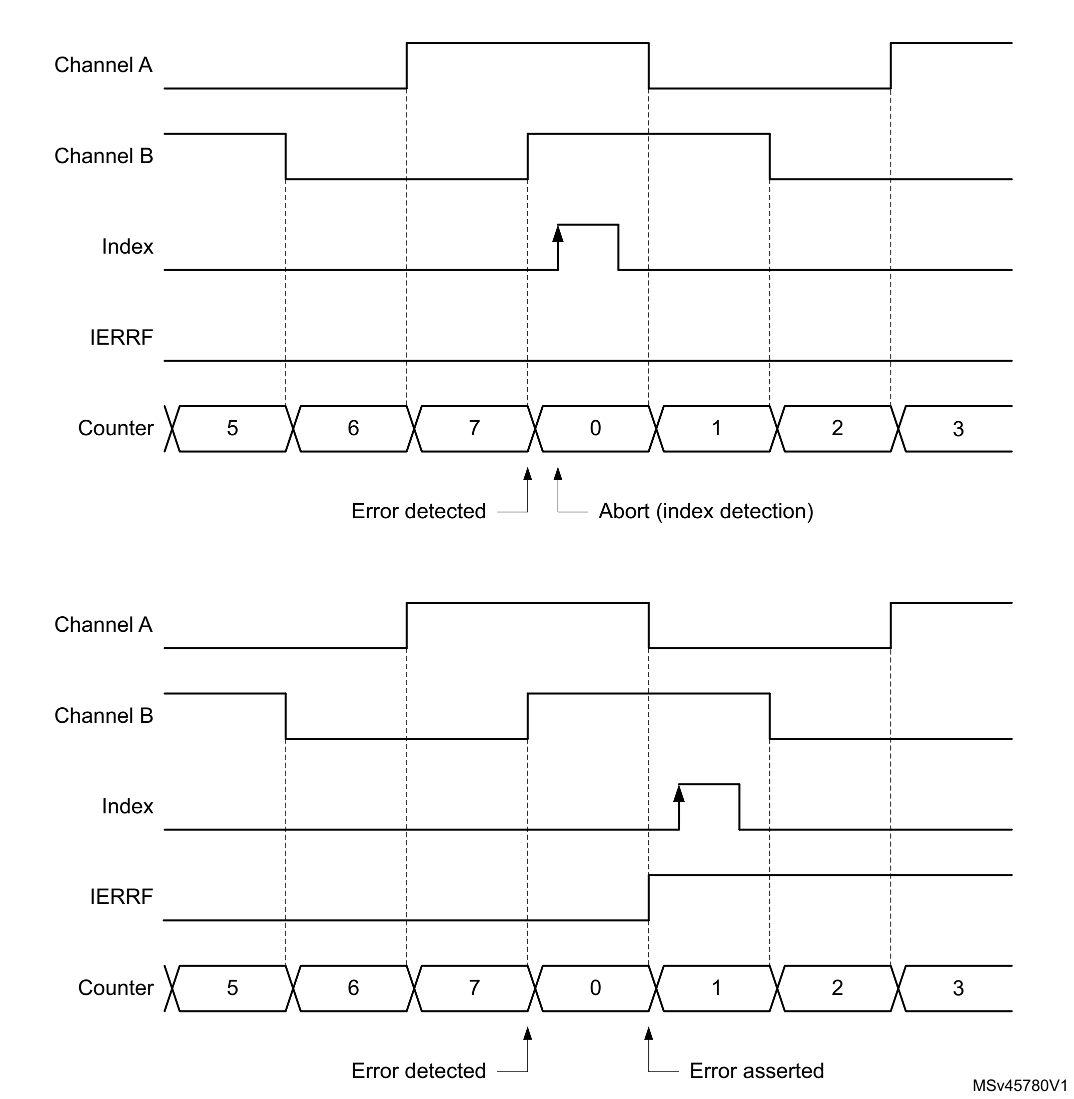
For encoder having an Index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The Index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an Index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

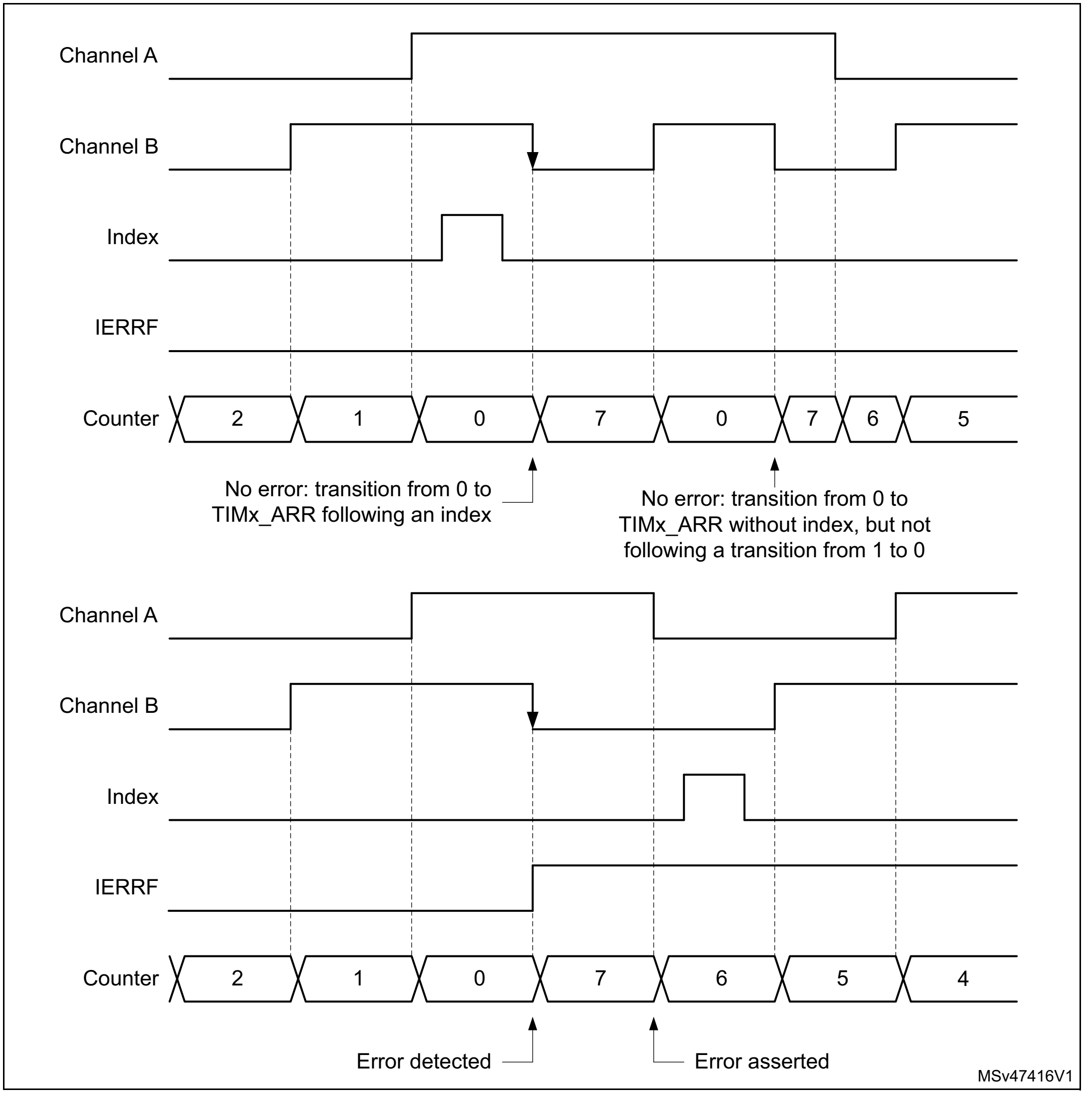
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 425* below.

Figure 425. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 426* below, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 426. Down-counting encode error detection


Channel A
Channel B
Index
IERRF
Counter
2
1
0
7
0
7
6
5
No error: transition from 0 to TIMx_ARR following an index
No error: transition from 0 to TIMx_ARR without index, but not following a transition from 1 to 0
Channel A
Channel B
Index
IERRF
Counter
2
1
0
7
6
5
4
Error detected
Error asserted
MSv47416V1

An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder Interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the Index event sets the IDXF interrupt flag in the TIMx_SR status register. An Index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

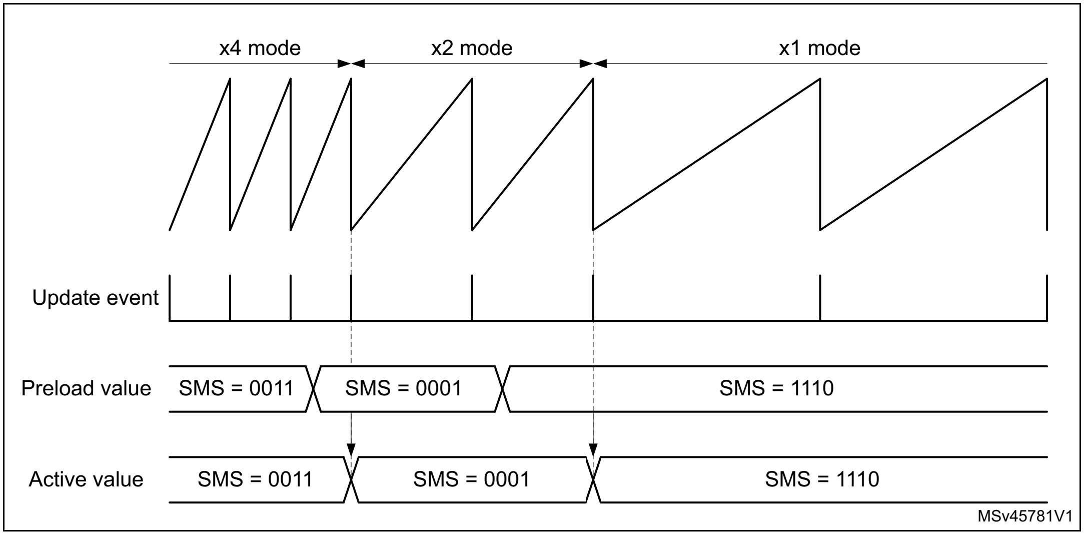
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the Update interrupt rate, by switching from x4 to x2 to x1 mode, as show on the *Figure 427* below.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting. This mode must be used only when index is disabled (bit IE = 0).
- SMSPS = 1: the transfer is triggered by the Index event.

Figure 427. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

29.4.19 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bit field to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

29.4.20 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 allows 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).

29.4.21 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 tim_ti1, tim_ti2 and tim_ti3.

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 21.3.25: Interfacing with Hall sensors on page 576*.

29.4.22 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

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.

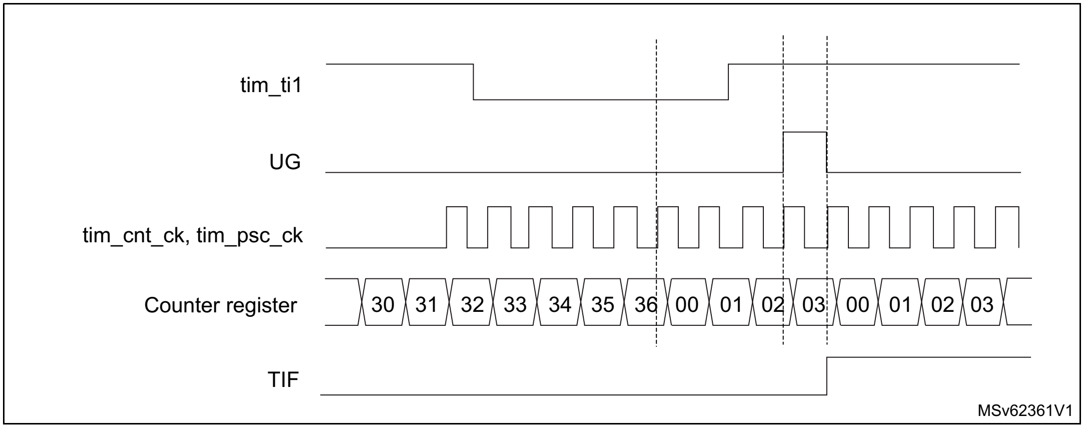
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_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 tim_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 tim_ti1 rising edge. When tim_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 tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 428. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

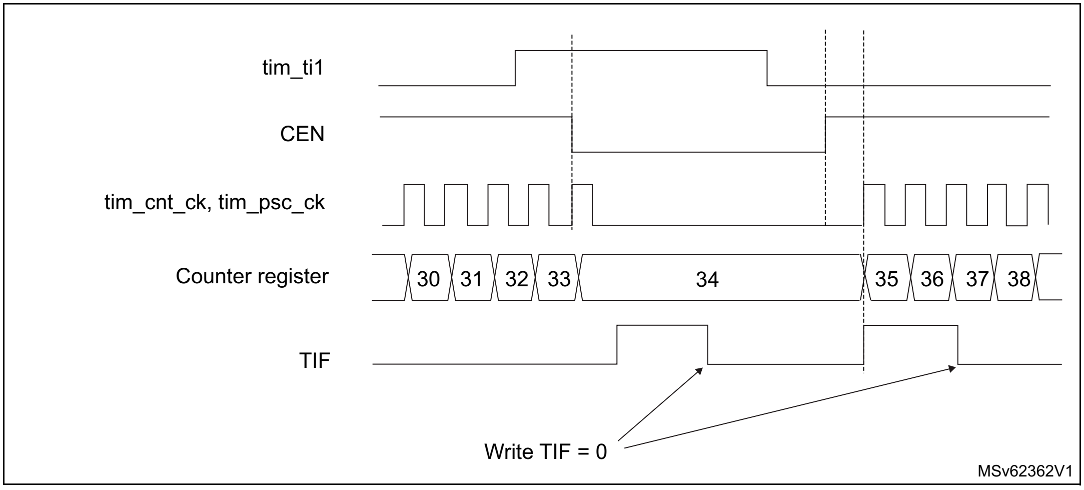
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_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 tim_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 tim_ti1 is low and stops as soon as tim_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 tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 429. Control circuit in gated mode



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.

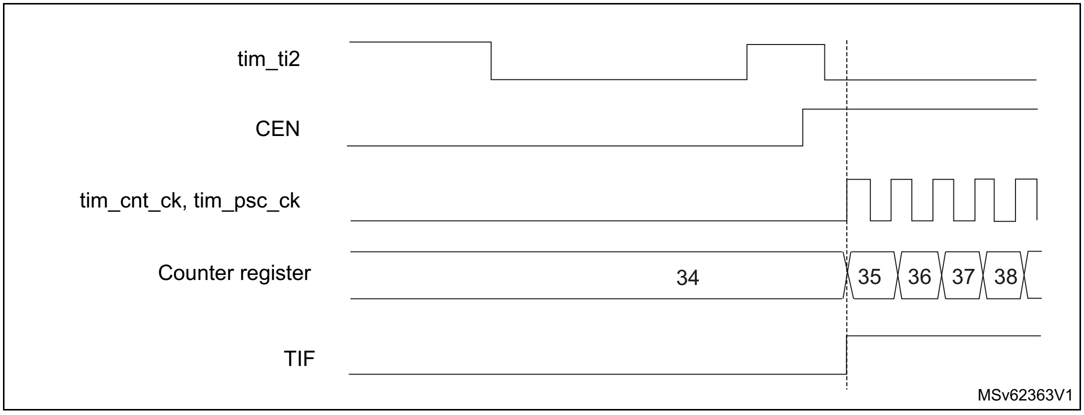
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_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 tim_ti2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 430. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode allows to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

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 tim_etr_in 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 tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

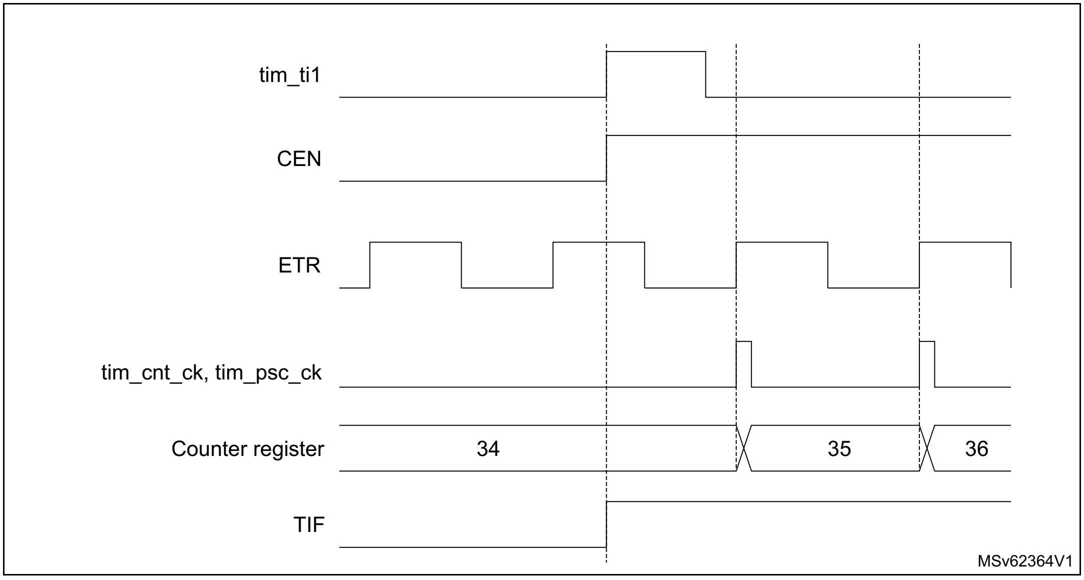
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

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 tim_etr_in 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 tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 431. Control circuit in external clock mode 2 + trigger mode



29.4.23 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 432 and *Figure 433* show examples of master/slave timer connections.

Figure 432. Master/Slave timer example

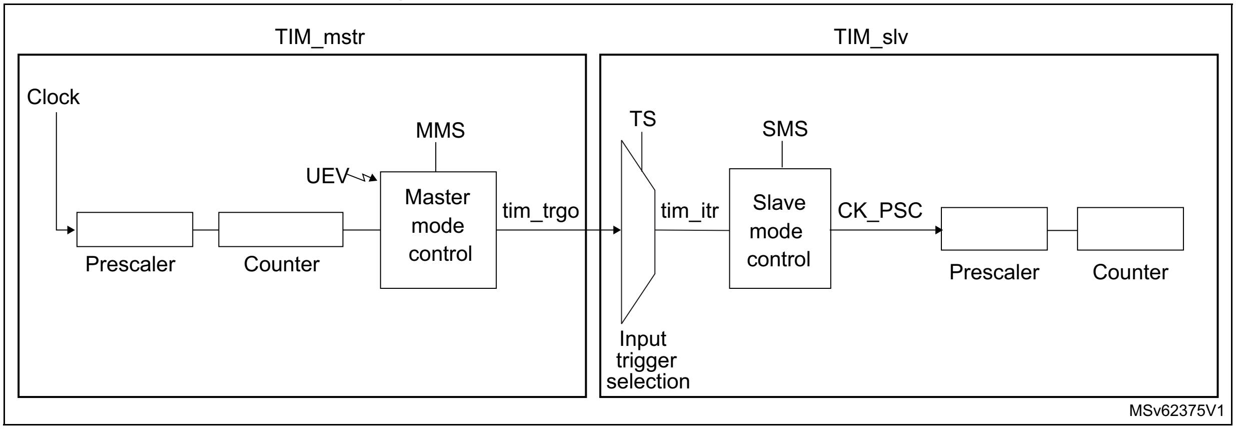
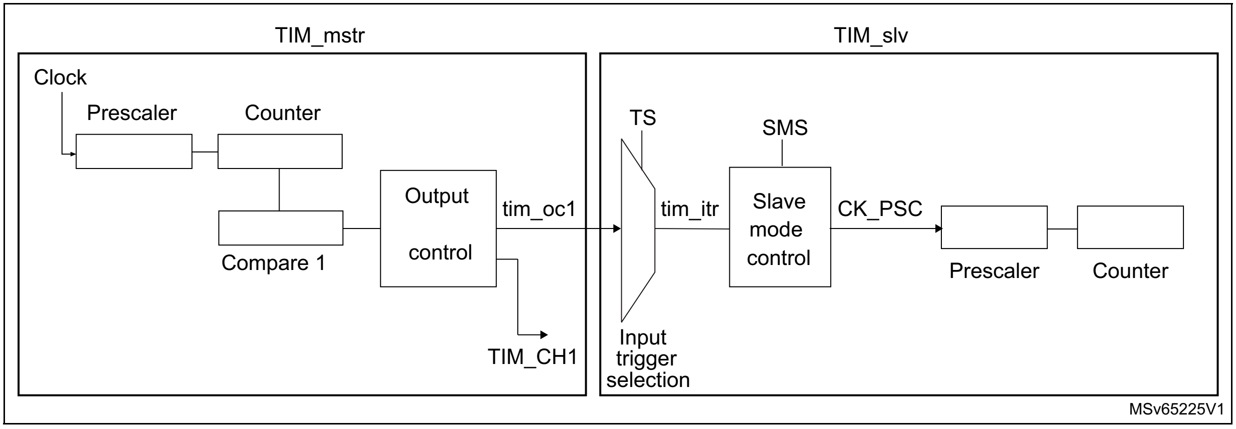


Figure 433. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 433) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 29.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals).
The tim_oc1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.
For instance, if the destination timer tim_ker_ck 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, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 432*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS=00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM_slv.

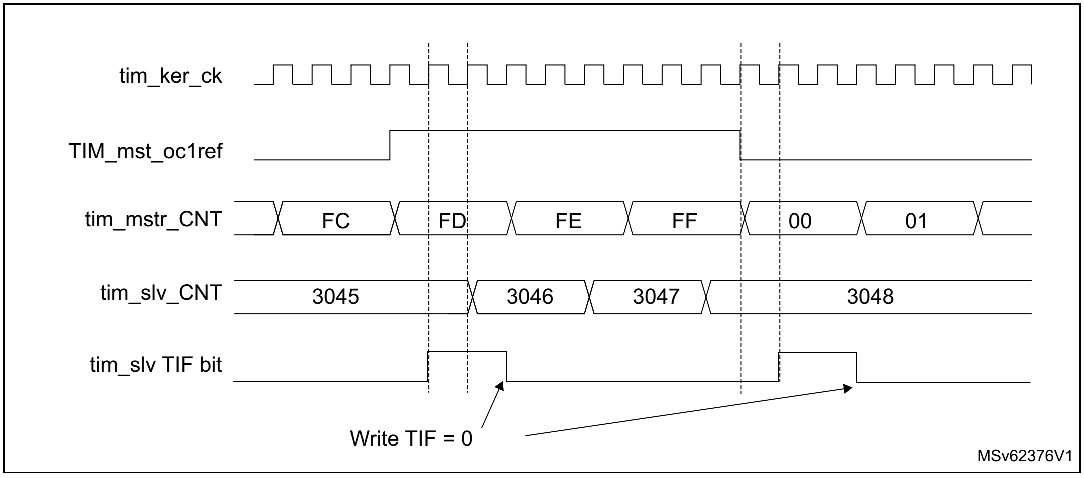
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 432* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim\_cnt\_ck} = f_{tim\_ker\_ck}/3$).

1. Configure TIM_mstr master mode to send its Output Compare 1 Reference (tim_oc1ref) signal as trigger output (MMS=100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS=101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing ‘1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing ‘1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 434. Gating TIM_slv with tim_oc1ref of TIM_mstr

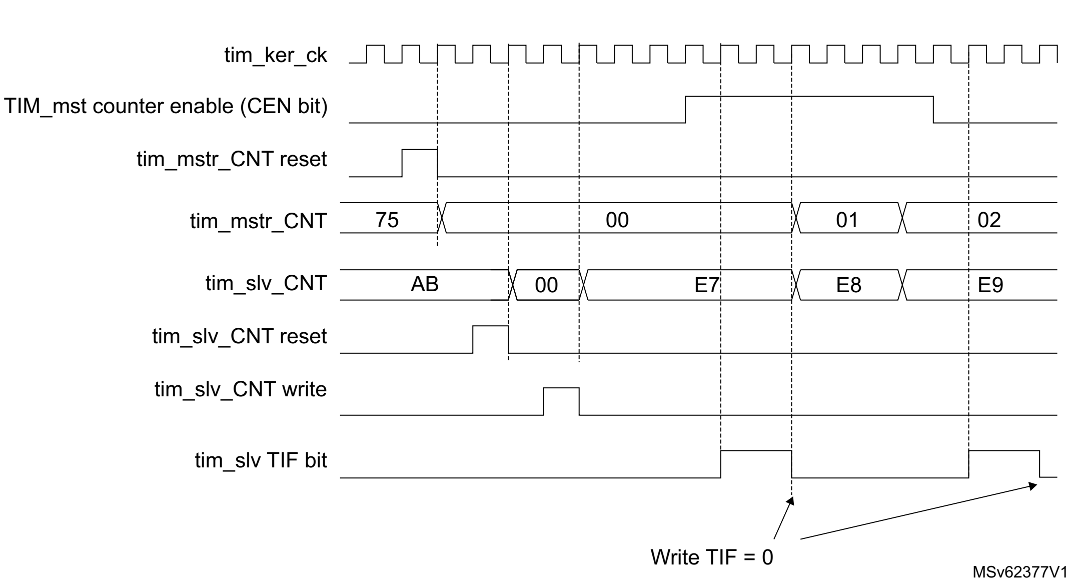


In the example in *Figure 434*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 435*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing ‘0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its Output Compare 1 Reference (tim_oc1ref) signal as trigger output (MMS=100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS=101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing '1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing '1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing '0xE7' in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing '1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing '1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing '0 in the CEN bit (TIM_mstr_CR1 register).

Figure 435. Gating TIM_slv with Enable of TIM_mstr

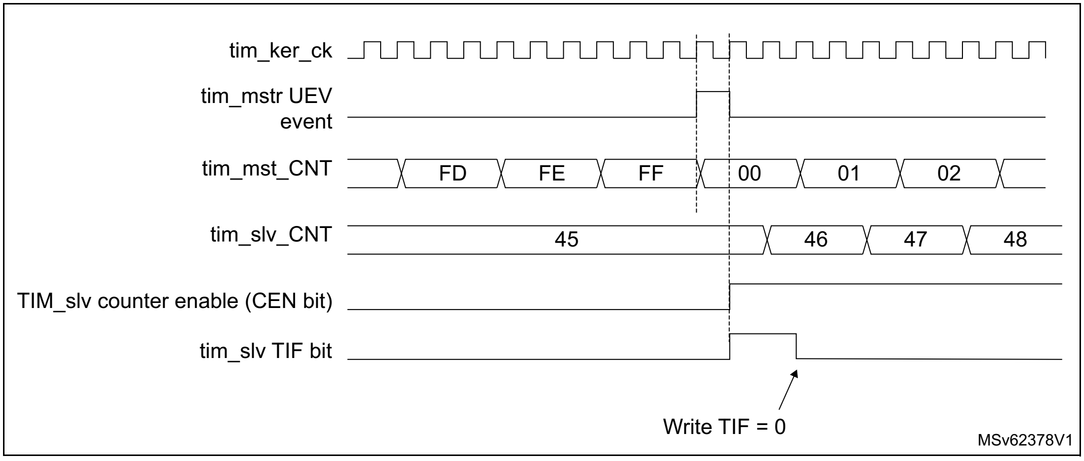


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 432* for connections. TIM_slv 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 TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write '0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim\_cnt\_ck} = f_{tim\_ker\_ck}/3$).

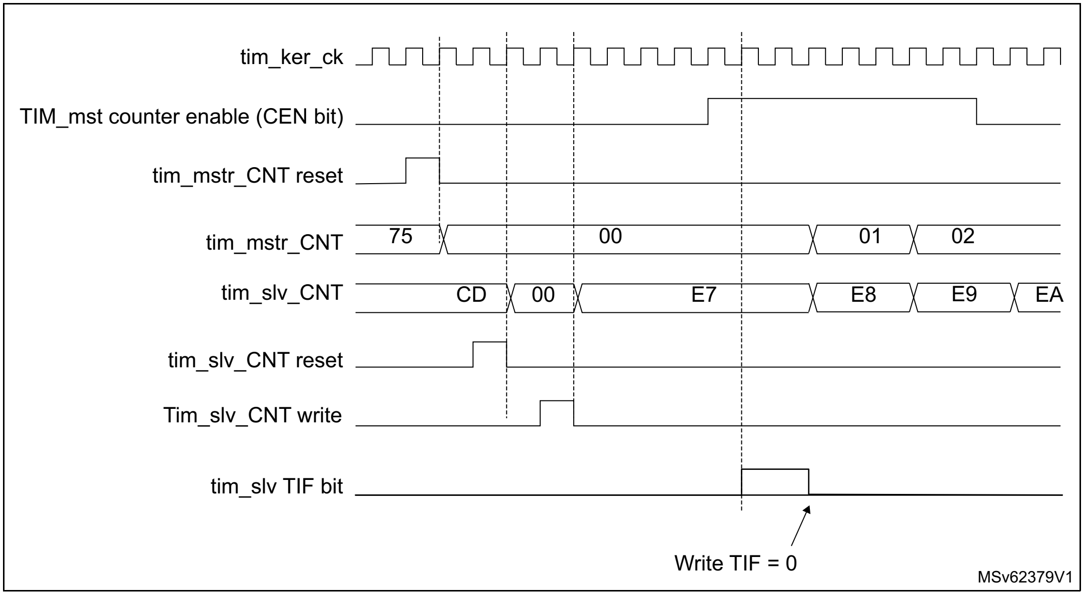
1. Configure TIM_mstr master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS=110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing '1 in the CEN bit (TIM_mstr_CR1 register).

Figure 436. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 437* shows the behavior with the same configuration as in *Figure 436* but in trigger mode instead of gated mode (SMS=110 in the TIM_slv_SMCR register).

Figure 437. Triggering TIM_slv with Enable of TIM_mstr



Starting 2 timers synchronously in response to an external trigger

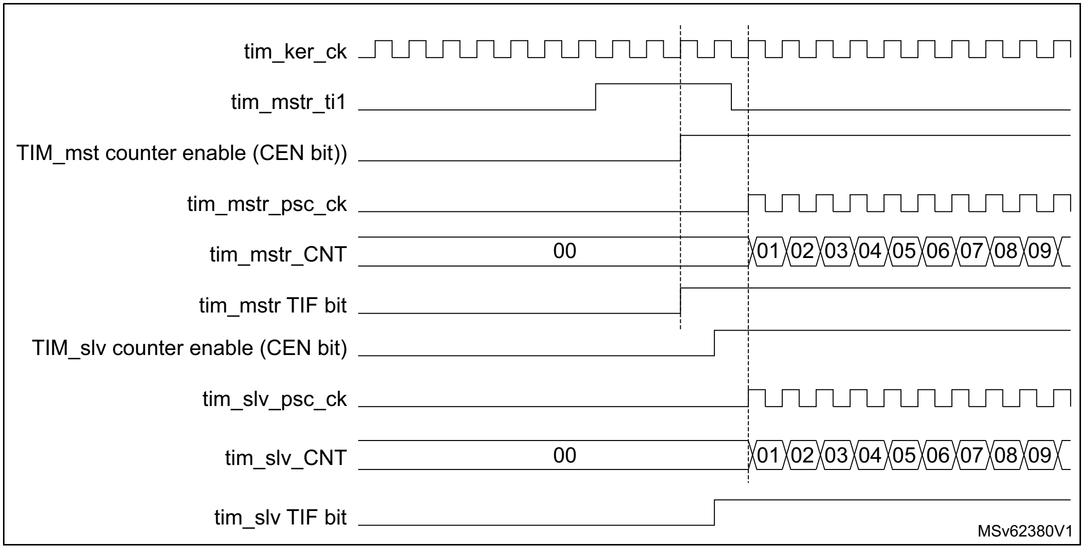
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 432* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its Enable as trigger output (MMS=001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS=00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS=110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM=1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS=110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 438. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_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.

29.4.24 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.

29.4.25 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in *Table 281*.

Table 281. DMA request

DMA acronym	DMA request	Enable control bit
TIM_UP	Update	UDE
TIM_CH1	Capture/compare 1	CC1DE
TIM_CH2	Capture/compare 2	CC2DE

Table 281. DMA request (continued)

DMA acronym	DMA request	Enable control bit
TIM_CH3	Capture/compare 3	CC3DE
TIM_CH4	Capture/compare 4	CC4DE
TIM_TRIG	Trigger	TDE

29.4.26 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU core halted), the TIMx counter can either continues to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

29.4.27 TIM2/TIM3/TIM4/TIM5 low-power modes

Table 282. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

29.4.28 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in *Table 283*.

Table 283. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRG	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No

Table 283. Interrupt requests (continued)

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

29.5 TIM2/TIM3/TIM4/TIM5 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).

29.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITH EN	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

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 (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim\_ker\_ck}$
01: $t_{DTS} = 2 \times t_{tim\_ker\_ck}$
10: $t_{DTS} = 4 \times t_{tim\_ker\_ck}$
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.

29.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

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.	MMS[3]	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.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15..0] multiplexer output is to tim_ti1 input
1: The tim_ti1_in[15..0], tim_ti2_in[15..0] and tim_ti3_in[15..0] multiplexers outputs are XORed and connected to the tim_ti1 input. See also *Section 21.3.25: Interfacing with Hall sensors on page 576*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_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 tim_trgo, 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 (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **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 (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

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.

29.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

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.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			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 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer’s Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 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 tim_etr_in or $\overline{\text{tim\_etr\_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in 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 tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_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, tim_trgi must not be connected to tim_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 tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF

01: tim_etrp frequency divided by 2

10: tim_etrp frequency divided by 4

11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_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: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, 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 (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_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 (see bits 21:20 for TS[4:3])

This bit-field selects the trigger input to be used to synchronize the counter.
00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
Others: Reserved

See *Section 29.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation details.

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: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to '0'. Section 29.3: TIM2/TIM3/TIM4/TIM5 implementation.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_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 tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_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 tim_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 (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_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.

29.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

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.	TERRIE	IERRIE	DIRIE	IDXIE	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.	TDE	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	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable

0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable

0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable

0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable

0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 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 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.

29.5.5 TIMx status register (TIMx_SR)(x = 2 to 5)

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.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
								rc_w0	rc_w0	rc_w0	rc_w0				
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 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to ‘0’.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to ‘0’.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag

This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to ‘0’.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag

This flag is set by hardware when an index event is detected. It is cleared by software by writing it to ‘0’.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 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 tim_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 downcounting 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.

29.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x014

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).

29.5.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

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).

31	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

Input capture mode

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, tim_ic2 is mapped on tim_ti2.
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_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 tim_ti1 input and the length of the digital filter applied to tim_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: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, 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 (tim_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, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_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).

29.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

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).

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

Output compare mode

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, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_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: tim_oc1ref is not affected by the tim_ocref_clr_int input
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

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. tim_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. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_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 (tim_oc1ref=0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_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 tim_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 tim_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 tim_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 tim_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 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: In PWM mode, the tim_ocref_clr level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

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: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

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, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_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).

29.5.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

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).

31	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

Input capture mode

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, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_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, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_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).

29.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

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).

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

Output compare mode

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, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_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, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_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).

29.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x020

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 tim_ti1fp1/tim_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 284. Output control bit for standard tim_ocx channels

CCxE bit	tim_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 tim_ocx channels depends only on the GPIO registers when CCxE=0.

29.5.12 TIMx counter (TIMx_CNT)(x = 3, 4)

Address offset: 0x024

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.
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**: Value depends on IUFREMAP in TIMx_CR1.

If UIFREMAP = 0

Reserved

If UIFREMAP = 1

UIFCPY: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value‘

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register holds the non-dithered part in CNT[15:0]. The fractional part is not available.

29.5.13 TIMx counter (TIMx_CNT)(x = 2, 5)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT [31] or UIF CPY	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 **CNT[31] or UIFCPY**: Value depends on IUFREMAP in TIMx_CR1.

If UIFREMAP = 0

CNT[31]: Most significant bit of counter value

If UIFREMAP = 1

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]**: Least significant part of counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

29.5.14 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x028

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 tim_cnt_ck is equal to $f_{tim\_psc\_ck}$ / (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”).

29.5.15 TIMx auto-reload register (TIMx_ARR)(x = 3, 4)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

31	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.	ARR[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Low Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 29.4.3: Time-base unit on page 1236* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

29.5.16 TIMx auto-reload register (TIMx_ARR)(x = 2, 5)

Address offset: 0x02C

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 29.4.3: Time-base unit on page 1236* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

29.5.17 TIMx capture/compare register 1 (TIMx_CCR1)(x = 3, 4)

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.	CCR1[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: 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 tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR1[15:0] bits hold the capture value. The CCR1[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:0]. The CCR1[3:0] bits are reset.

29.5.18 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2, 5)

Address offset: 0x034

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 tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

29.5.19 TIMx capture/compare register 2 (TIMx_CCR2)(x = 3, 4)

Address offset: 0x038

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.	CCR2[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 1 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_CCMR2 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 tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR2[15:0] bits hold the capture value. The CCR2[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:0]. The CCR2[3:0] bits are reset.

29.5.20 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2, 5)

Address offset: 0x038

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_CCMR2 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 tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

29.5.21 TIMx capture/compare register 3 (TIMx_CCR3)(x = 3, 4)

Address offset: 0x03C

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.	CCR3[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare 3 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_CCMR3 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 signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR3[15:0] bits hold the capture value. The CCR3[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[19:0]. The CCR3[3:0] bits are reset.

29.5.22 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2, 5)

Address offset: 0x03C

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 3 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_CCMR3 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 signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

29.5.23 TIMx capture/compare register 4 (TIMx_CCR4)(x = 3, 4)

Address offset: 0x040

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.	CCR4[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare 4 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_CCMR4 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 signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR4[15:0] bits hold the capture value. The CCR4[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:0]. The CCR4[3:0] bits are reset.

29.5.24 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2, 5)

Address offset: 0x040

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 4 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_CCMR4 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 signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

29.5.25 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5)

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.	PWPRSC[2:0]			PW[7:0]							
					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.	IPOS[1:0]		FIDX	Res.	Res.	IDIR[1:0]		IE
								rw	rw	rw			rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Note: The IDR[1:0] bitfield must be written when IE bit is reset (index disabled).

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

29.5.26 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)

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.	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 tim_ti4[0..15] input

0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 29.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[0..15] input

0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 29.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[0..15] input

0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 29.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[0..15] input

0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 29.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

29.5.27 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5)

Address offset: 0x060

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_in source selection

These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 29.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

29.5.28 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5)

Address offset: 0x064

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.	OCRSEL[2:0]		
													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 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
...
111: tim_ocref_clr7
Refer to *Section 29.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

29.5.29 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x3DC

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.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 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
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_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 are 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 are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are 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,
...

29.5.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x3E0

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).

29.5.31 TIMx register map

TIMx registers are mapped as described in the table below.

Table 285. TIM2/TIM3/TIM4/TIM5 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	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	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	0
0x004	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	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	0			
0x008	TIMx_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	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	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	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		0		0	0	0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	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	0	0	0	0
0x014	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
0x018	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
	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
0x01C	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
	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
0x020	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 285. TIM2/TIM3/TIM4/TIM5 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
0x024	TIMx_CNT	CNT[31] or UIFCPY	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[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
0x028	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
0x02C	TIMx_ARR	ARR[31:20] (32-bit timers only)												ARR[19:0]																			
	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
0x030	Reserved																																
0x034	TIMx_CCR1	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	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
0x038	TIMx_CCR2	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	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	TIMx_CCR3	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	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	TIMx_CCR4	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	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.. 0x054	Reserved	Res.																															
0x058	TIMx_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	Res.	Res.	IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0			0	0	0
0x05C	TIMx_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
0x060	TIMx_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.	Res.
	Reset value															0	0	0	0														
0x064	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	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
0x3E0	TIMx_DMAR	DMAB[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 2.2 on page 81* for the register boundary addresses.

30 General purpose timers (TIM15/TIM16/TIM17)

30.1 TIM15/TIM16/TIM17 introduction

The TIM15/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 TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 30.4.25: Timer synchronization (TIM15)*.

30.2 TIM15 main features

TIM15 includes 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
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- 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
- 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, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

30.3 TIM16/TIM17 main features

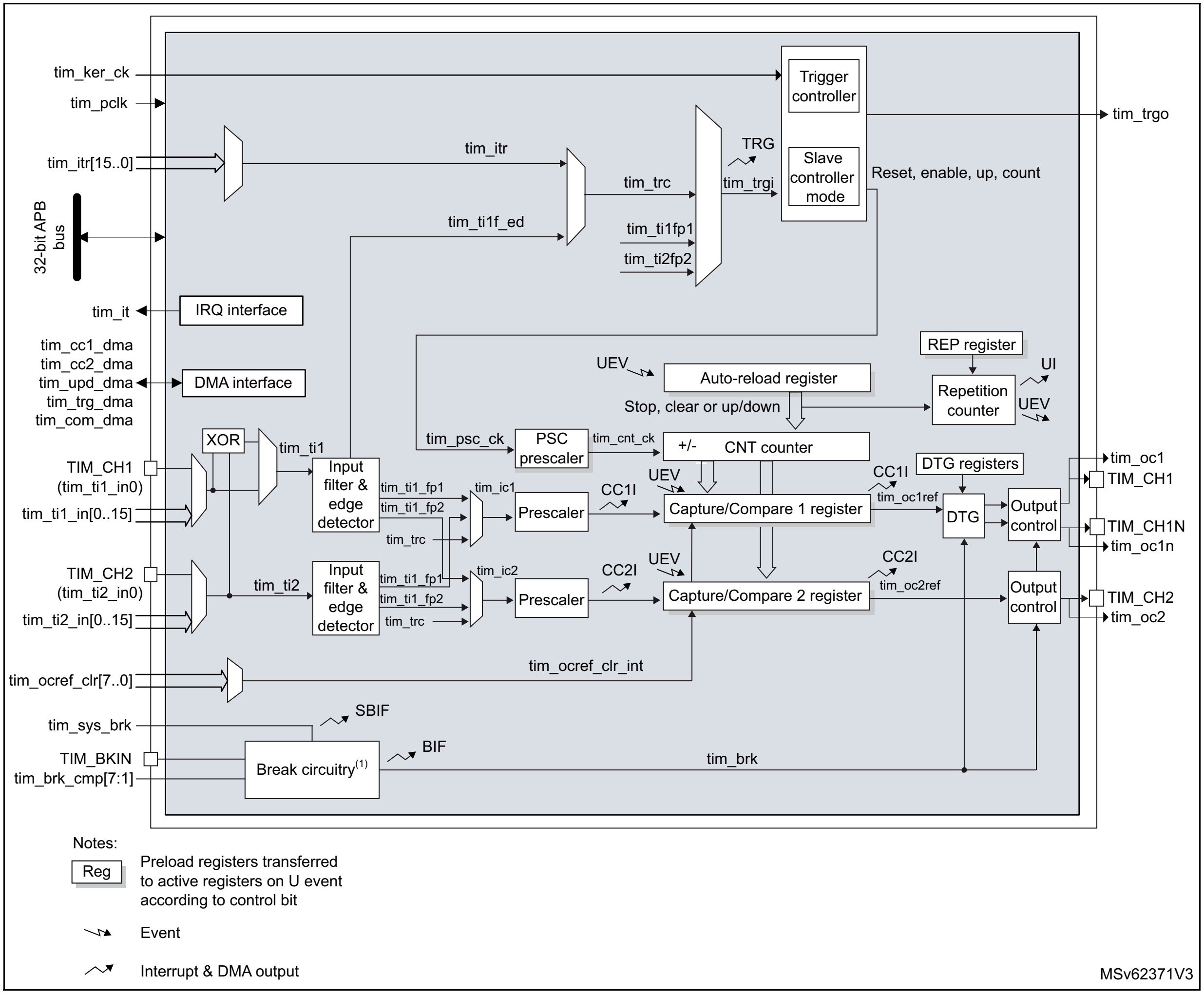
The TIM16/TIM17 timers include the following features:

- 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

30.4 TIM15/TIM16/TIM17 functional description

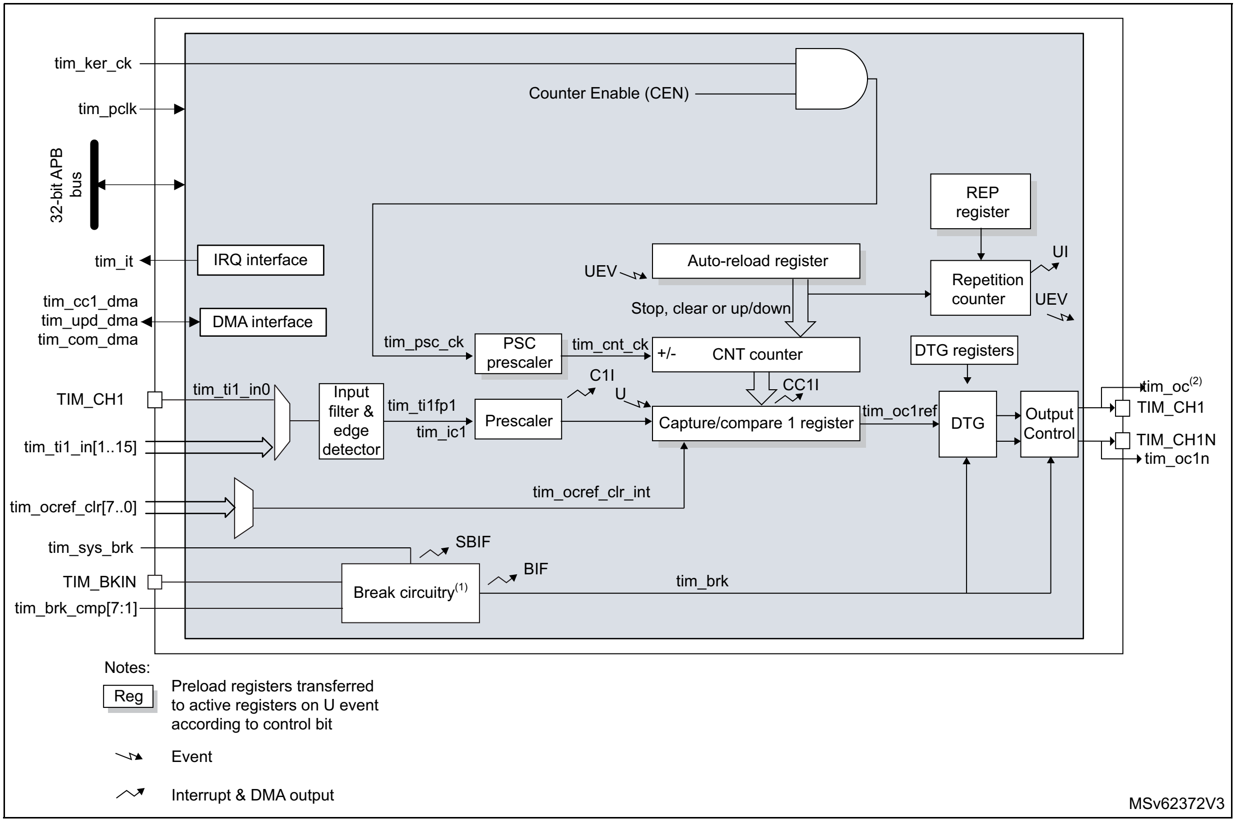
30.4.1 Block diagram

Figure 439. TIM15 block diagram



1. Refer to *Section 30.4.15: Using the break function* for details.

Figure 440. TIM16/TIM17 block diagram



1. Refer to *Section 30.4.15: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 30.4.26: Using timer output as trigger for other timers (TIM16/TIM17)* for details.

30.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 286 and *Table 287* in this section summarize the TIM inputs and outputs.

Table 286. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2[1]	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

1. Available for TIM15 only.

Table 287. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0][1]	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0][1]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo[1]	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[7:1]	Input	Break input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_upd_dma	Output	Timer update dma request
tim_trg_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

1. Available for TIM15 only.

Table 288 and *Table 289* list the sources connected to the tim_ti[1..2] input multiplexers.

Table 288. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in0	TIM15_CH1	TIM16_CH1	TIM17_CH1
tim_ti1_in1	LSE	comp6_out	comp5_out
tim_ti1_in2	comp1_out	MCO	MCO
tim_ti1_in3	comp2_out	HSE / 32[1]	HSE / 32[1]

Table 288. Interconnect to the tim_ti1 input multiplexer (continued)

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in4	comp5_out	RTC Clock	RTC Clock
tim_ti1_in5	comp7_out	LSE	LSE
tim_ti1_in6	Reserved	LSI	LSI
tim_ti1_in[7..15]	Reserved		

1. This signal is available only if the HSE32EN bit is set in the TIMx_OR1 register. See *Section 30.8.19: TIMx option register 1 (TIMx_OR1)(x = 16 to 17)* for details.

Table 289. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM15
tim_ti2_in0	TIM15_CH2
tim_ti2_in1	comp2_out
tim_ti2_in2	comp3_out
tim_ti2_in3	comp6_out
tim_ti2_in4	comp7_out
tim_ti2_in[5..15]	Reserved

Table 290 lists the internal sources connected to the tim_itr input multiplexer.

Table 290. TIMx internal trigger connection

tim_itrx inputs	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	tim5_trgo
tim_itr5	tim8_trgo
tim_itr6	Reserved
tim_itr7	tim16_oc1
tim_itr8	tim17_oc1
tim_itr9	tim20_trgo
tim_itr10	hrtim_out_sync2
tim_itr11..15	Reserved

Table 291 and *Table 292* list the sources connected to the tim_brk input.

Table 291. Timer break interconnect

tim_brk inputs	TIM15	TIM16	TIM17
TIM_BKIN	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin
tim_brk_cmp1	comp1_out	comp1_out	comp1_out
tim_brk_cmp2	comp2_out	comp2_out	comp2_out
tim_brk_cmp3	comp3_out	comp3_out	comp3_out
tim_brk_cmp4	comp4_out	comp4_out	comp4_out
tim_brk_cmp5	comp5_out	comp5_out	comp5_out
tim_brk_cmp6	comp6_out	comp6_out	comp6_out
tim_brk_cmp7	comp7_out	comp7_out	comp7_out

Table 292. System break interconnect

tim_sys_brk inputs	TIM15 / TIM16 / TIM17	Enable bit in SYSCFG_CFGR2 register
tim_sys_brk0	Cortex®-M4 with FPU LOCKUP	CLL
tim_sys_brk1	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk2	SRAM parity error	SPL
tim_sys_brk3	Flash double ECC error	ECCL
tim_sys_brk4	Clock Security System (CSS)	None (always enabled)

Table 293 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 293. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment		
	TIM15	TIM16	TIM17
tim_ocref_clr0	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out
tim_ocref_clr2	comp3_out	comp3_out	comp3_out
tim_ocref_clr3	comp4_out	comp4_out	comp4_out
tim_ocref_clr4	comp5_out	comp5_out	comp5_out
tim_ocref_clr5	comp6_out	comp6_out	comp6_out
tim_ocref_clr6	comp7_out	comp7_out	comp7_out
tim_ocref_clr7	Reserved		

30.4.3 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 tim_cnt_ck, 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 441 and *Figure 442* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 441. Counter timing diagram with prescaler division change from 1 to 2

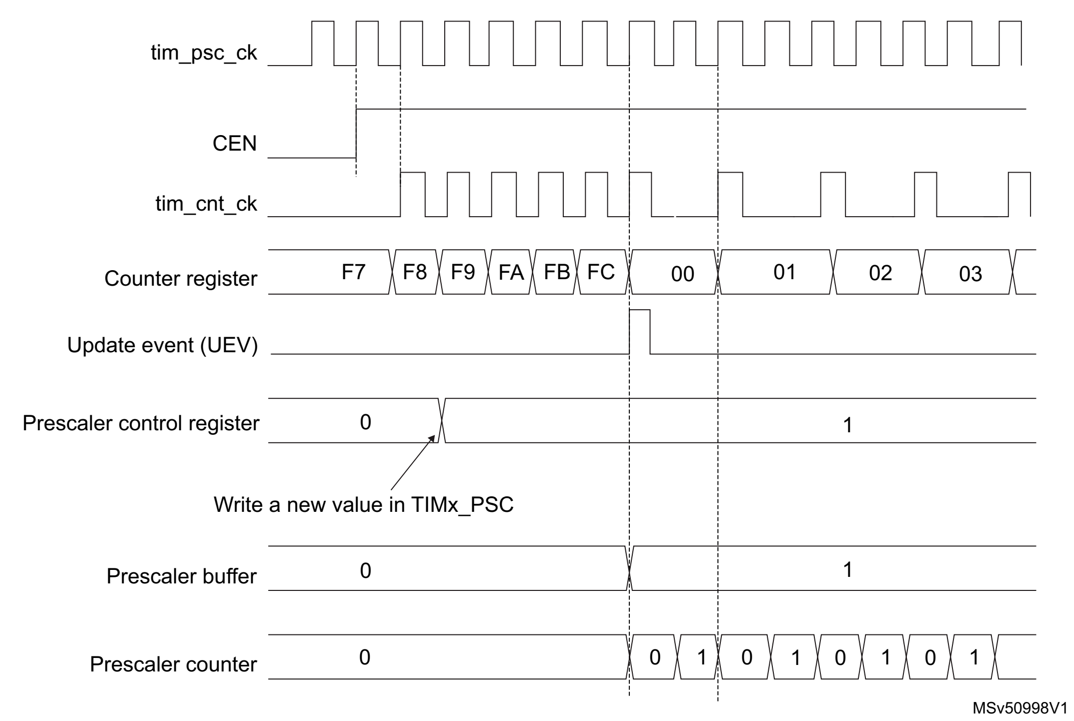
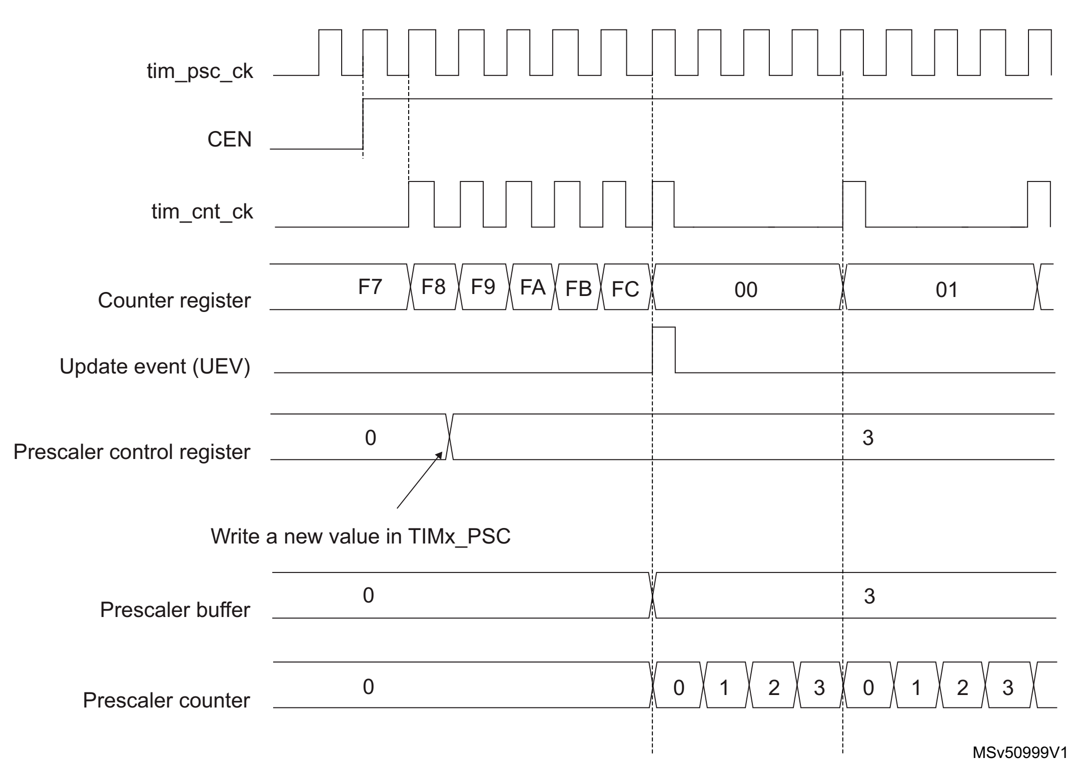


Figure 442. Counter timing diagram with prescaler division change from 1 to 4



30.4.4 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 443. Counter timing diagram, internal clock divided by 1

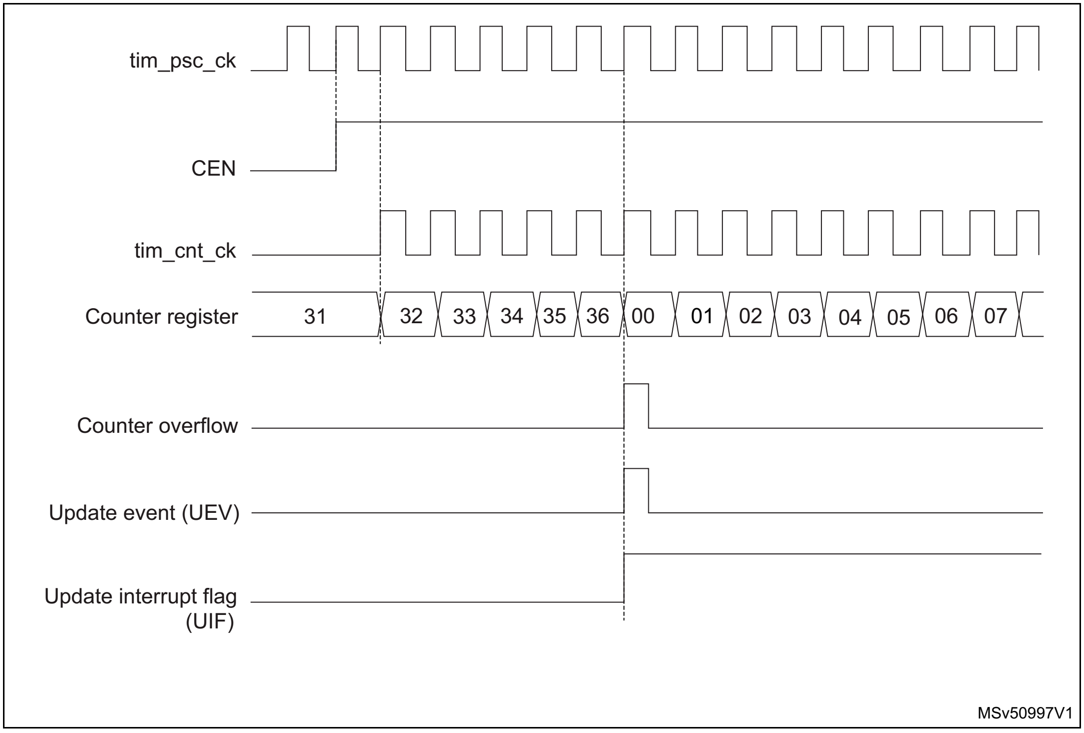


Figure 444. Counter timing diagram, internal clock divided by 2

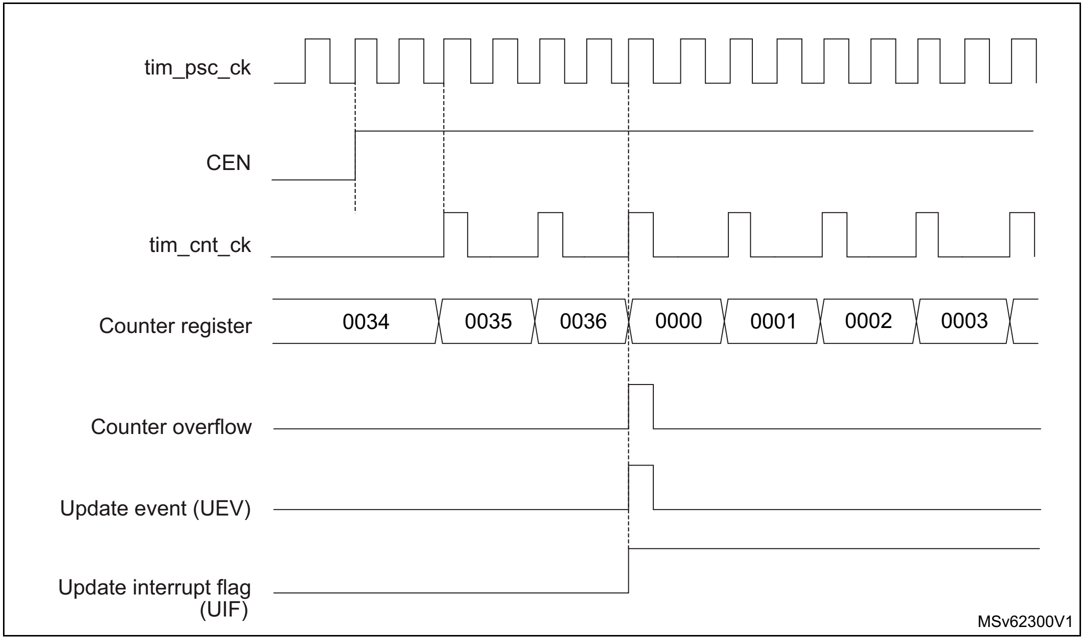


Figure 445. Counter timing diagram, internal clock divided by 4

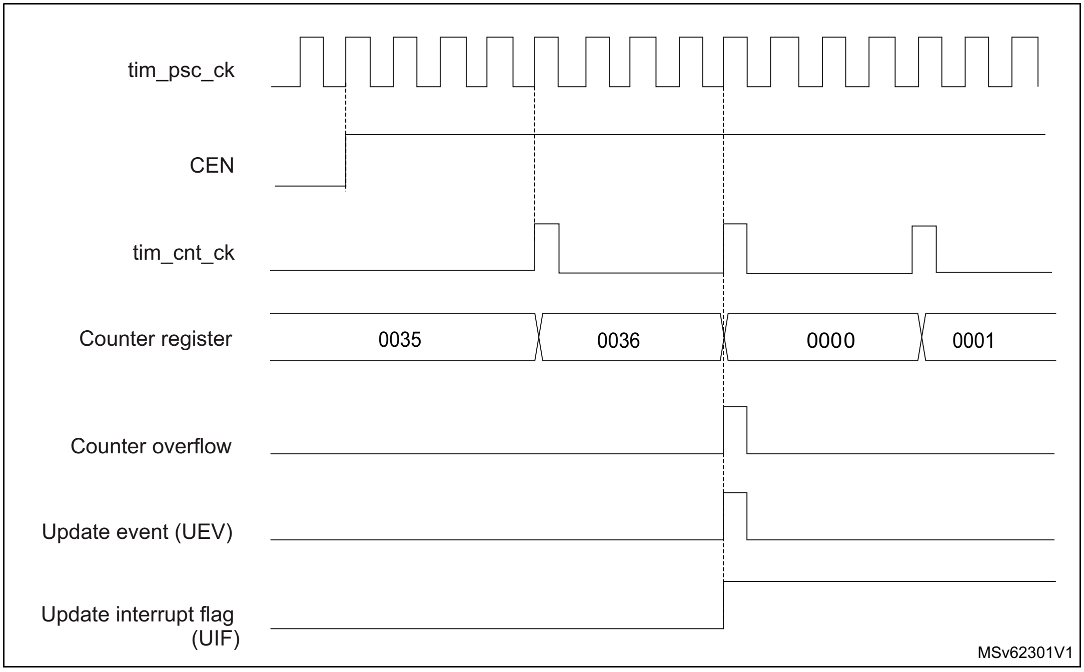


Figure 446. Counter timing diagram, internal clock divided by N

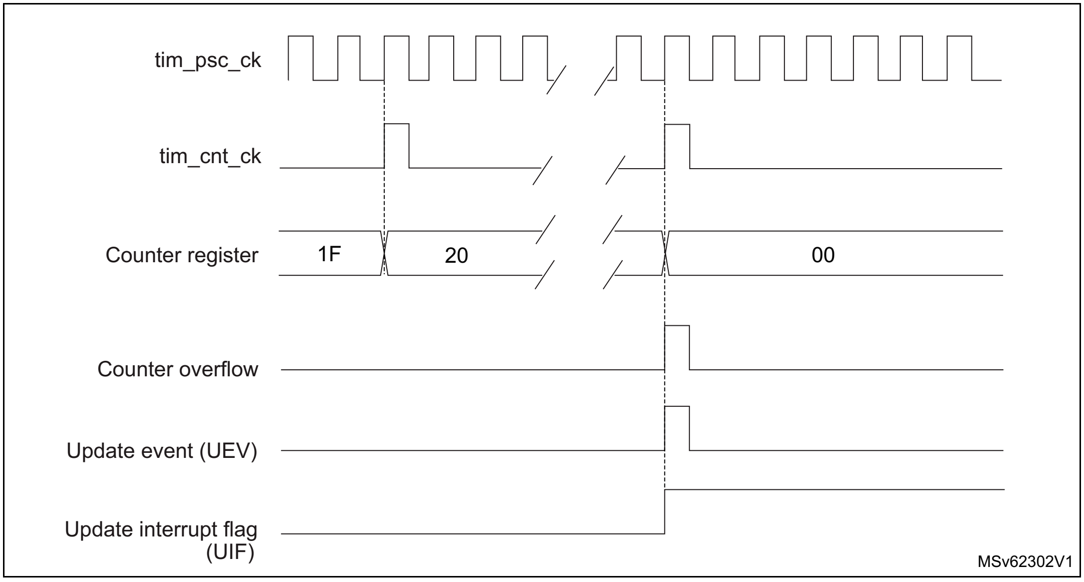


Figure 447. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

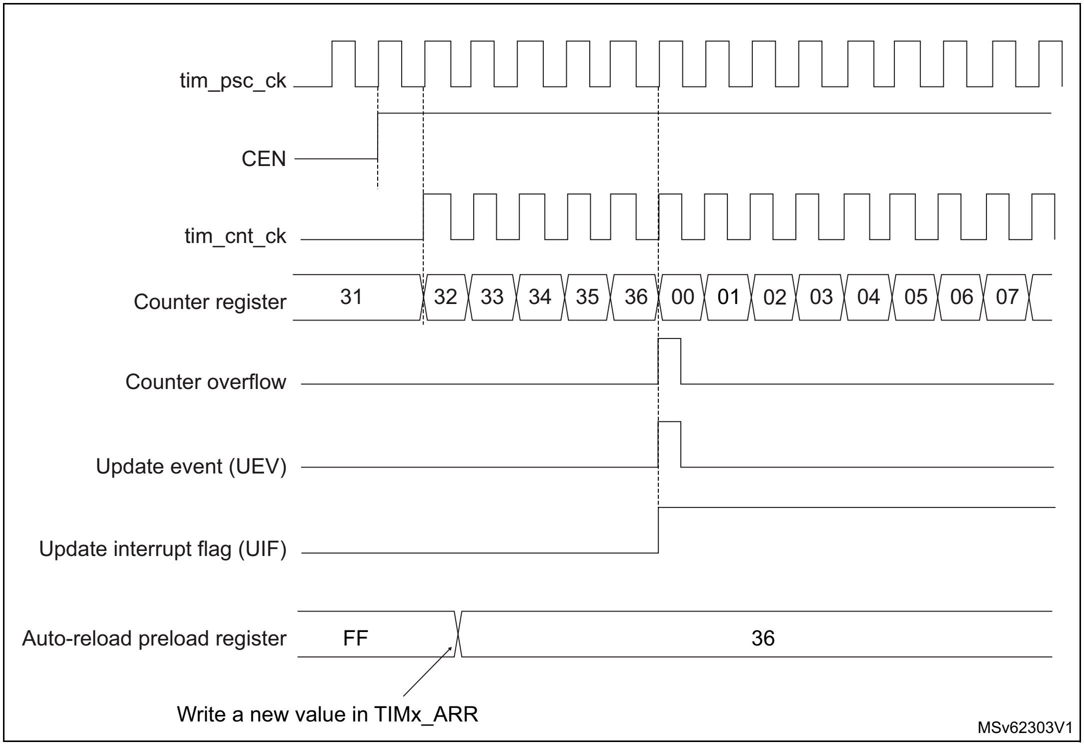
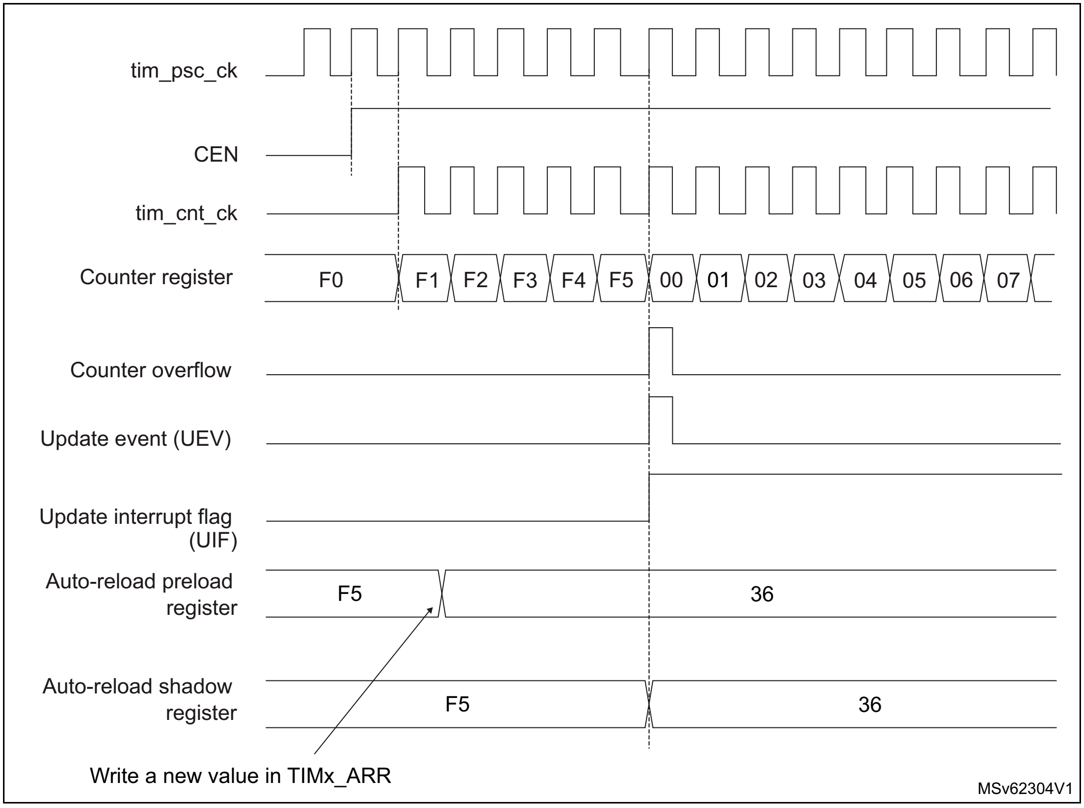


Figure 448. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



30.4.5 Repetition counter

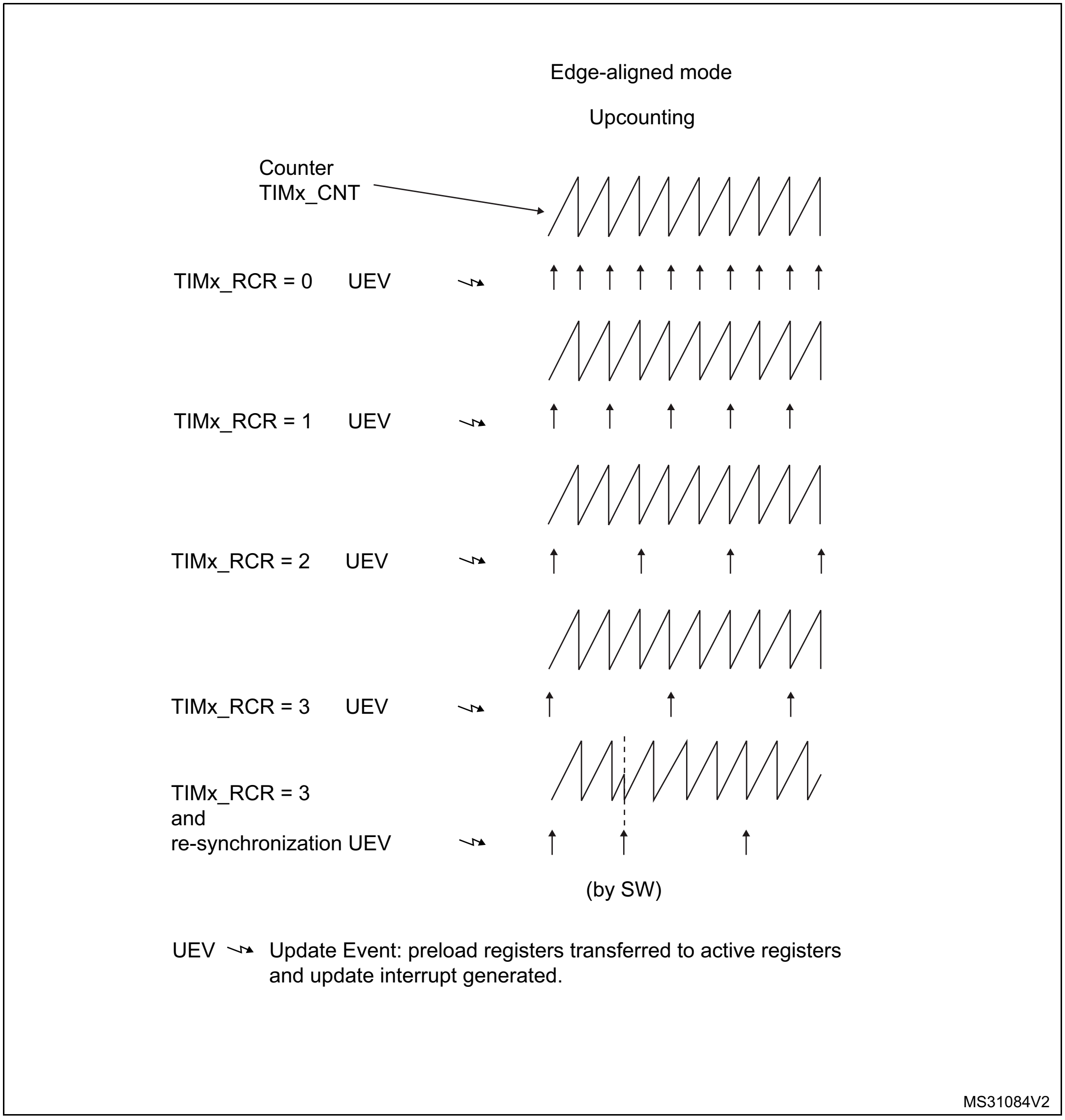
Section 30.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

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 449*). 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 449. Update rate examples depending on mode and TIMx_RCR register settings



30.4.6 Clock selection

The counter clock can be provided by the following clock sources:

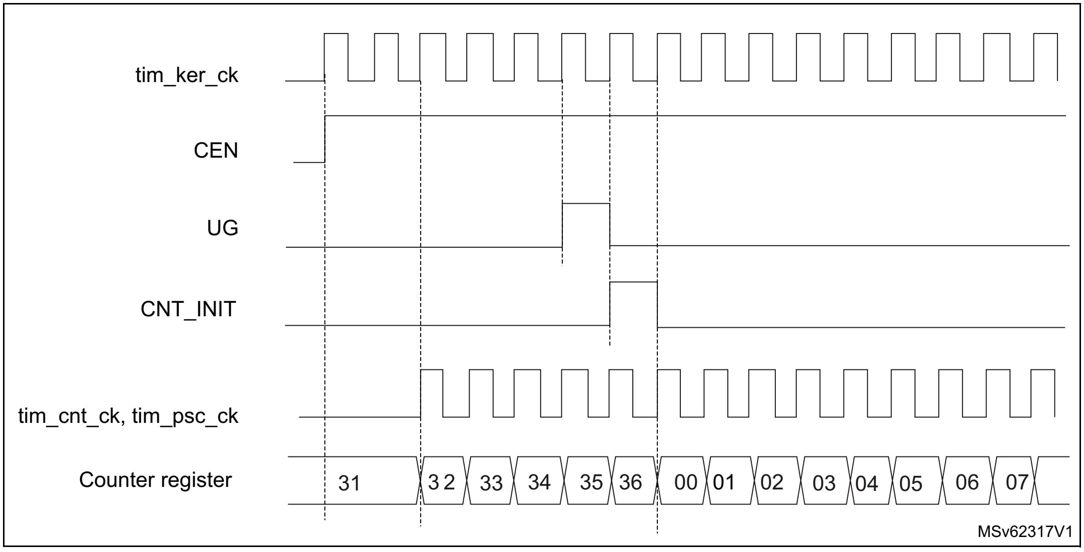
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer to enable another timer* for more details.

Internal clock source (tim_ker_ck)

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 tim_ker_ck.

Figure 450 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

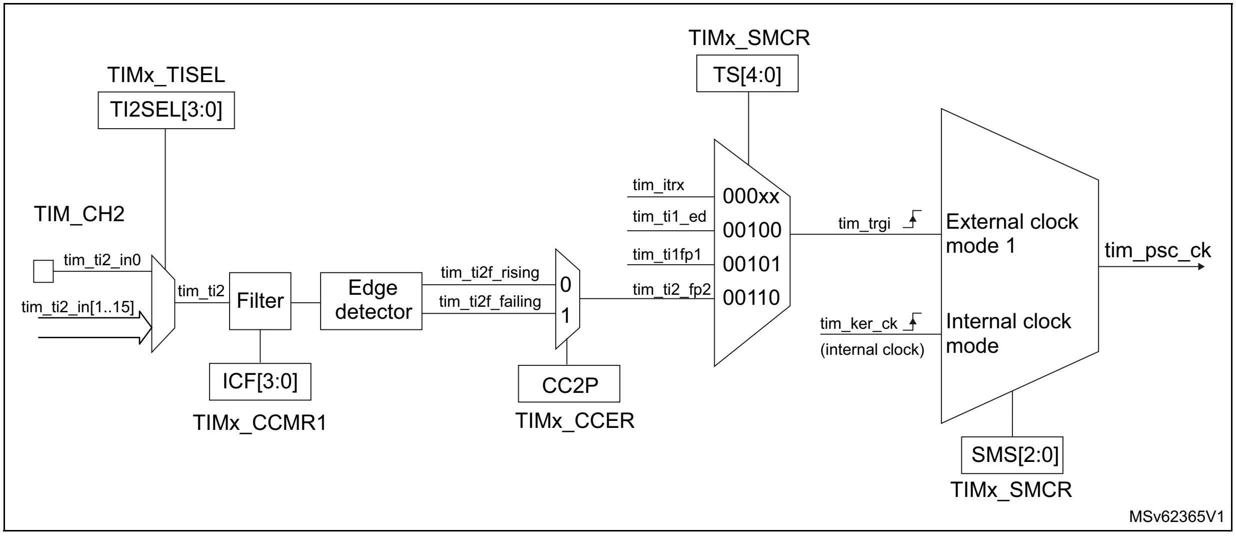
Figure 450. Control circuit in normal mode, internal clock divided by 1



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 451. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

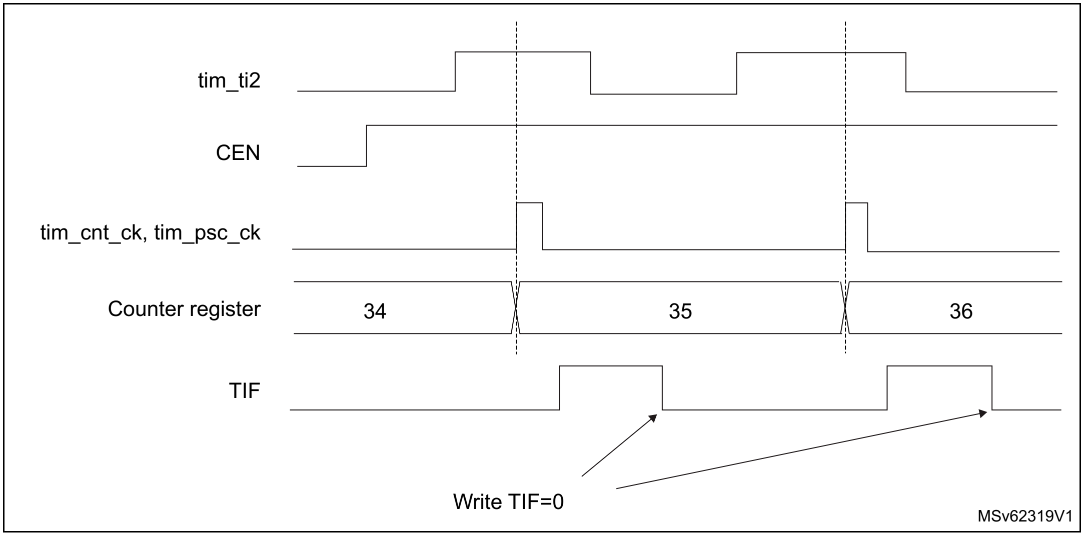
1. Select the proper tim_ti2_in[0..15] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] 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 tim_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, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 452. Control circuit in external clock mode 1



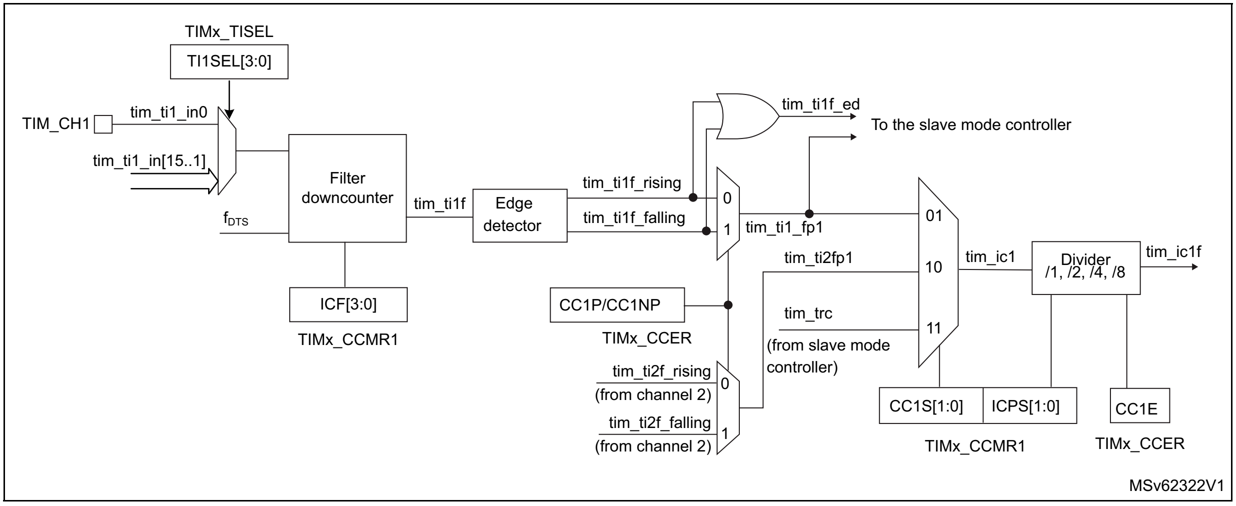
30.4.7 Capture/compare channels

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 453 to *Figure 456* give an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) 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 453. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 454. Capture/compare channel 1 main circuit

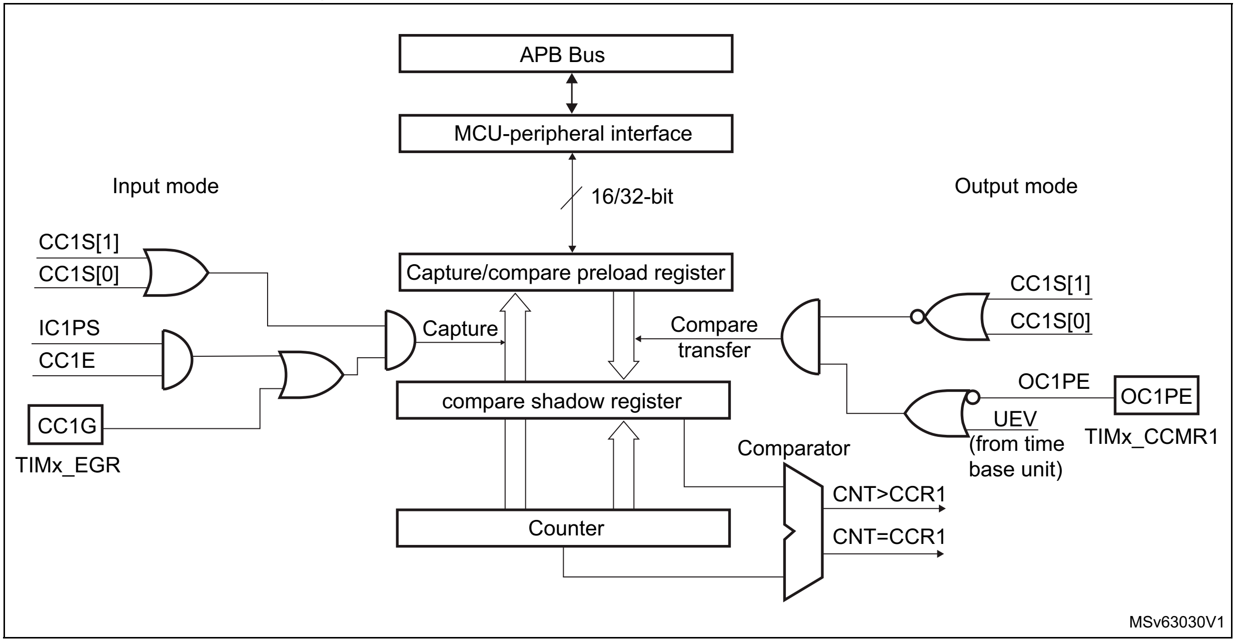


Figure 455. Output stage of capture/compare channel (channel 1)

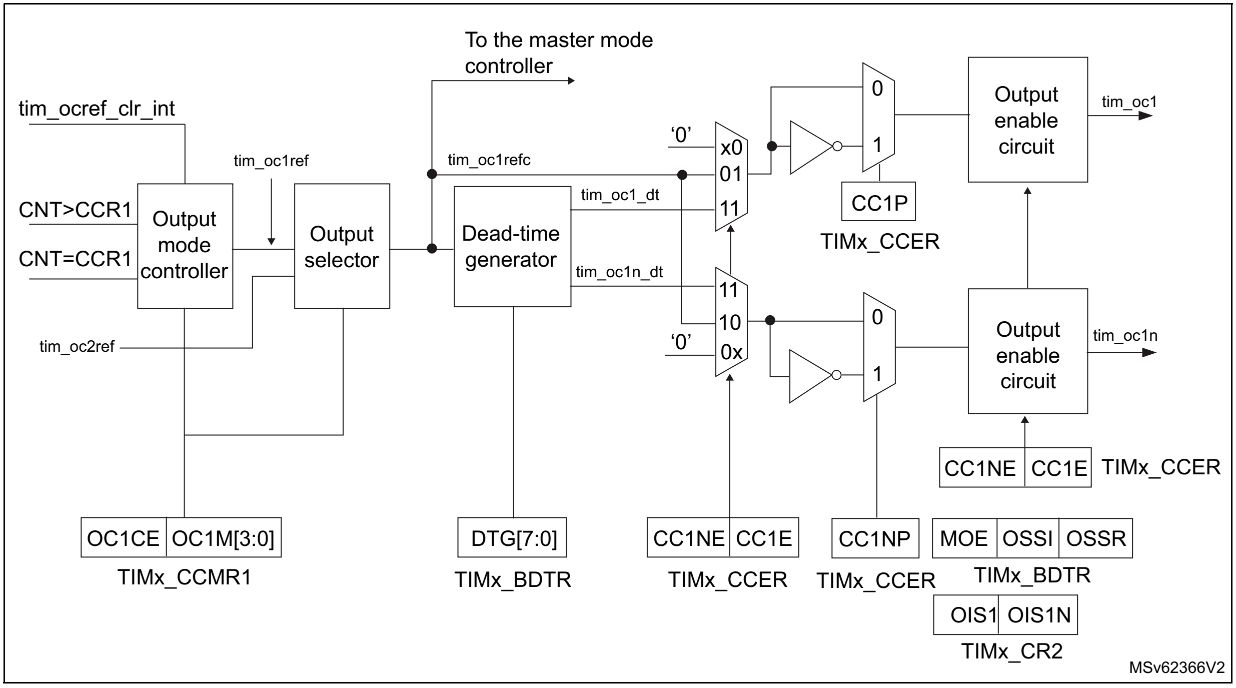
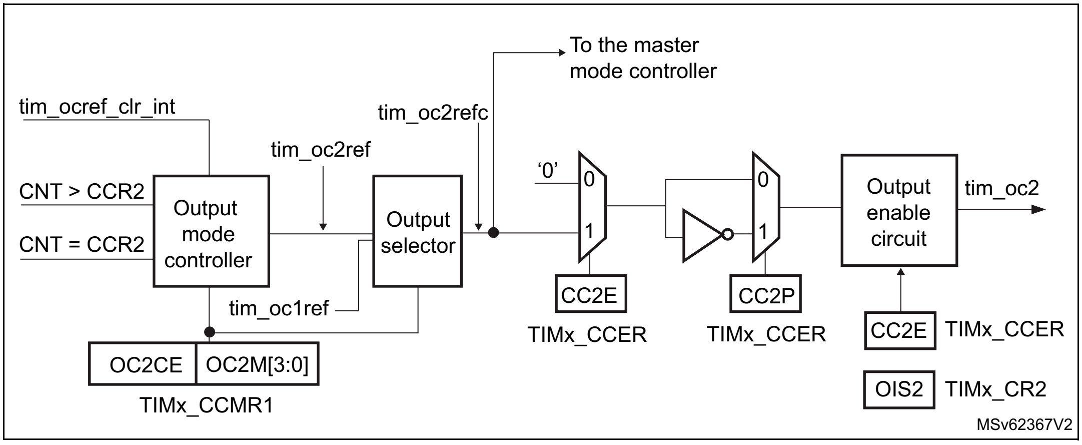


Figure 456. Output stage of capture/compare channel (channel 2 for TIM15)



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.

30.4.8 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 tim_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 tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[1..15] 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 tim_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 tim_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 tim_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 tim_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.

30.4.9 PWM input mode (only for TIM15)

This mode allows to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIMx_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges

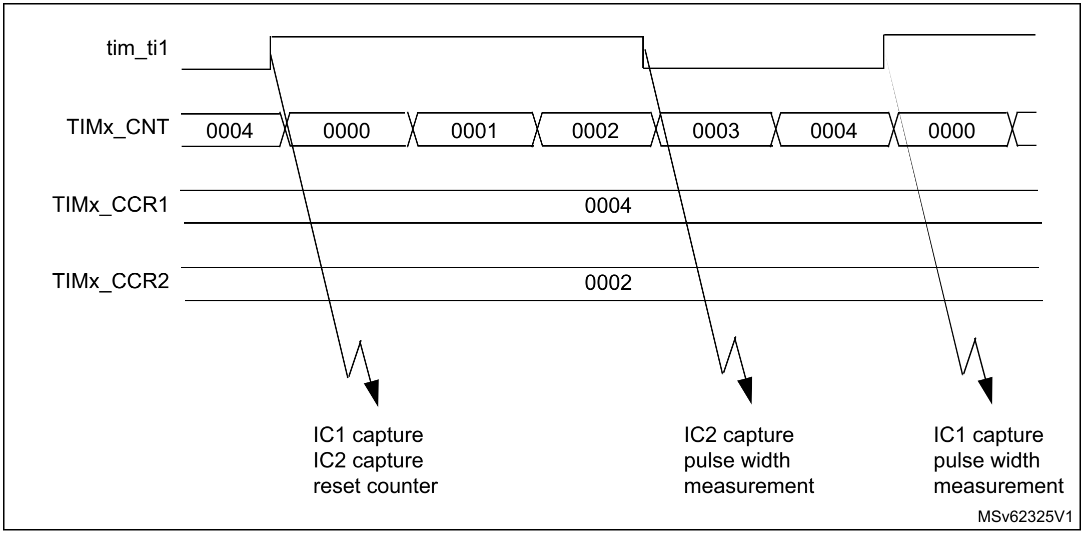
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These 2 tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy 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 tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[0.15] 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 (tim_ti1 selected).
3. Select the active polarity for tim_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 (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to ‘10’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_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 457. 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 tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

30.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_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 (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_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.

30.4.11 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.

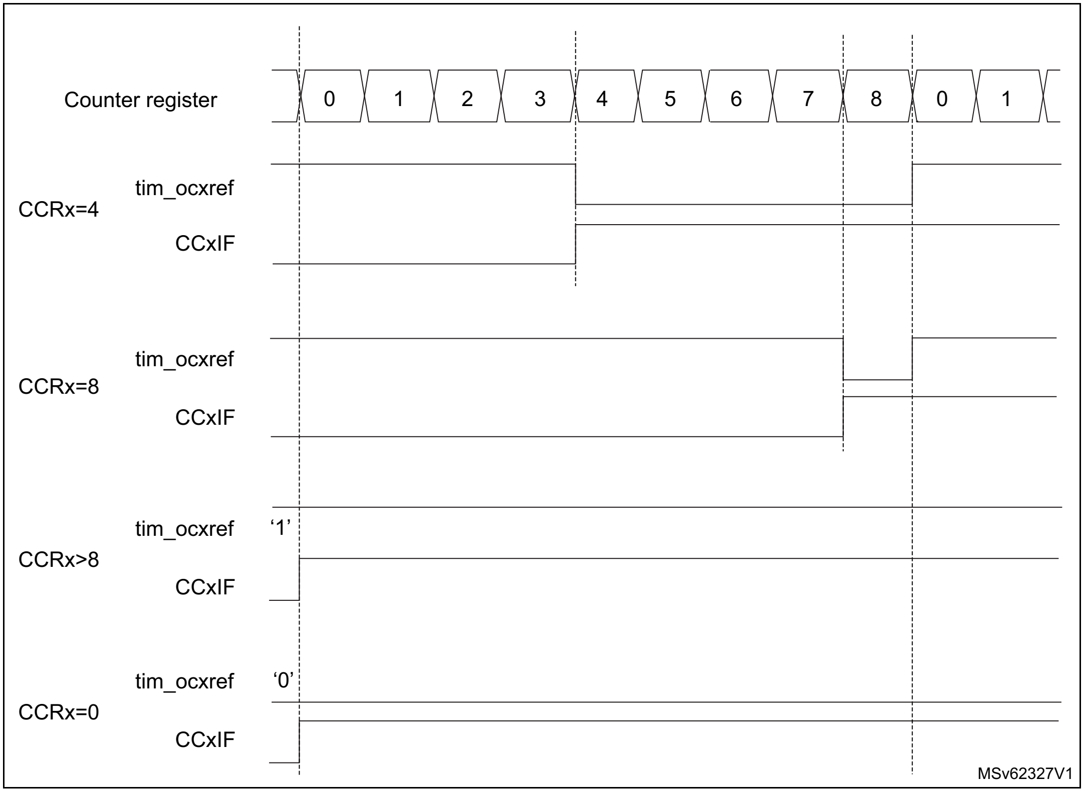
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 tim_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 458*.

Figure 458. Output compare mode, toggle on tim_oc1



30.4.12 PWM mode

Pulse Width Modulation mode allows 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 tim_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.

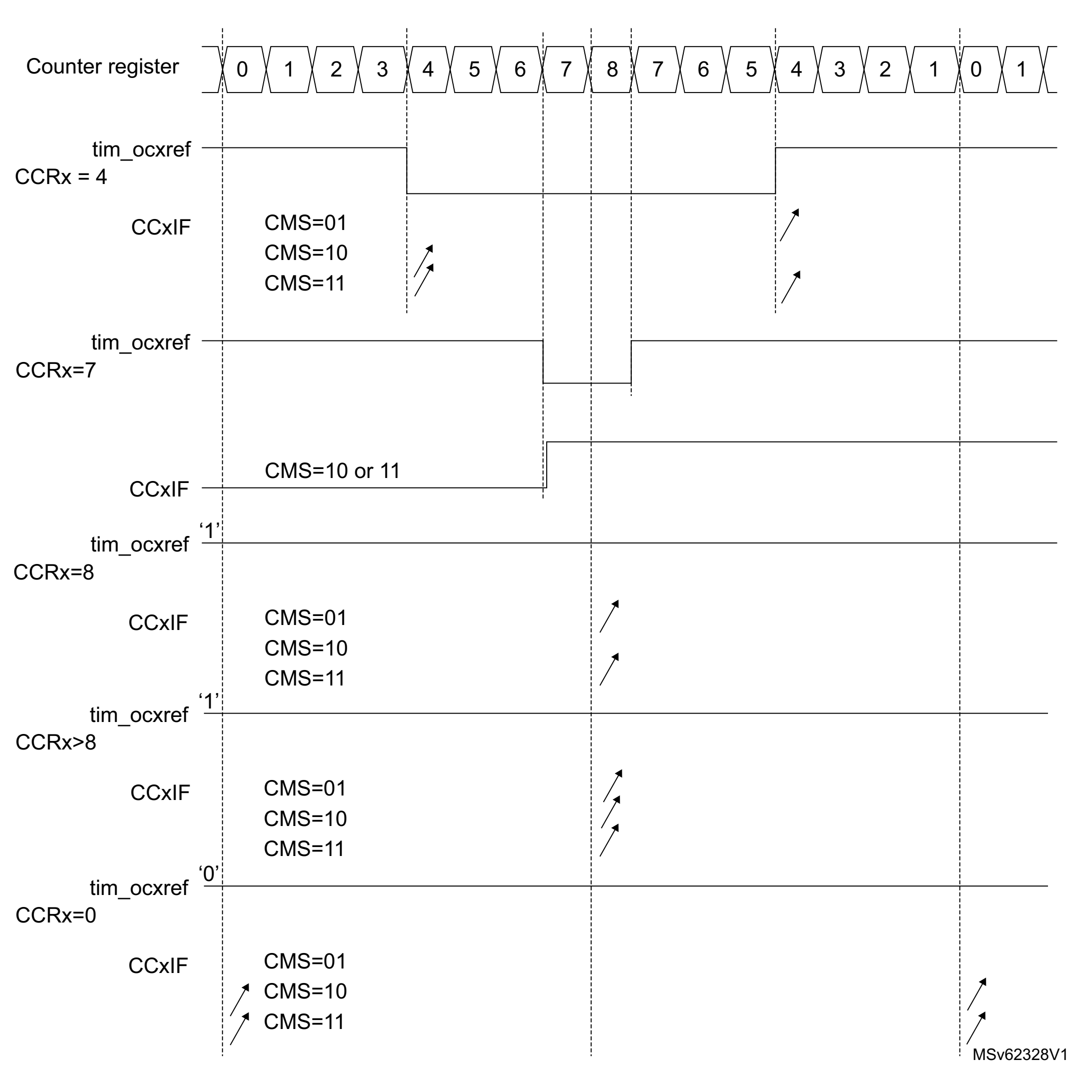
tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description 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 TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode on page 1355*.

In the following example, we consider PWM mode 1. The reference PWM signal tim_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 tim_ocxref is held at ‘1’. If the compare value is 0 then tim_ocxref is held at ‘0’. *Figure 459* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 459. Edge-aligned PWM waveforms (ARR=8)

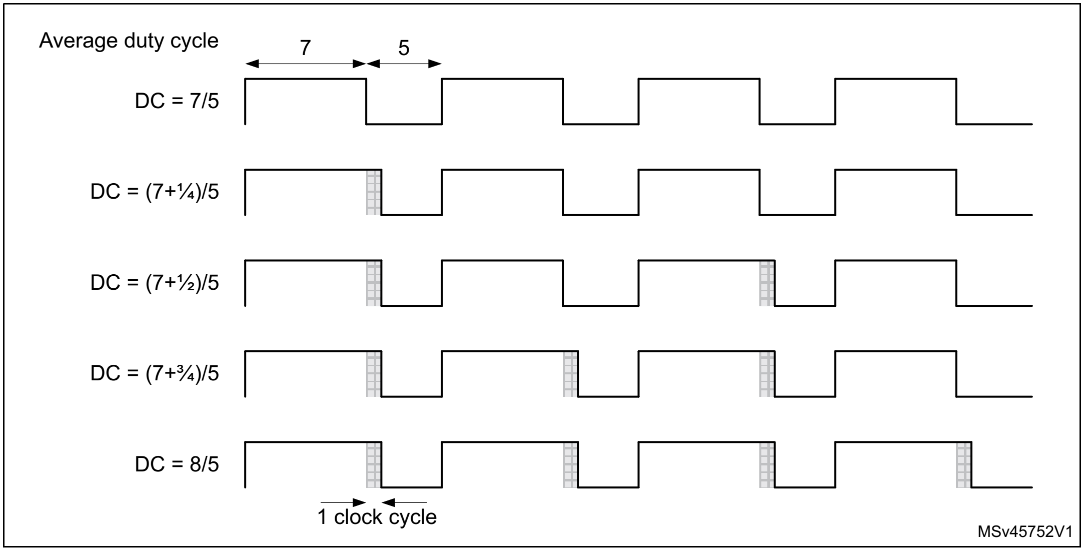


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 460* below presents the dithering principle applied to 4 consecutive PWM cycles.

Figure 460. Dithering principle

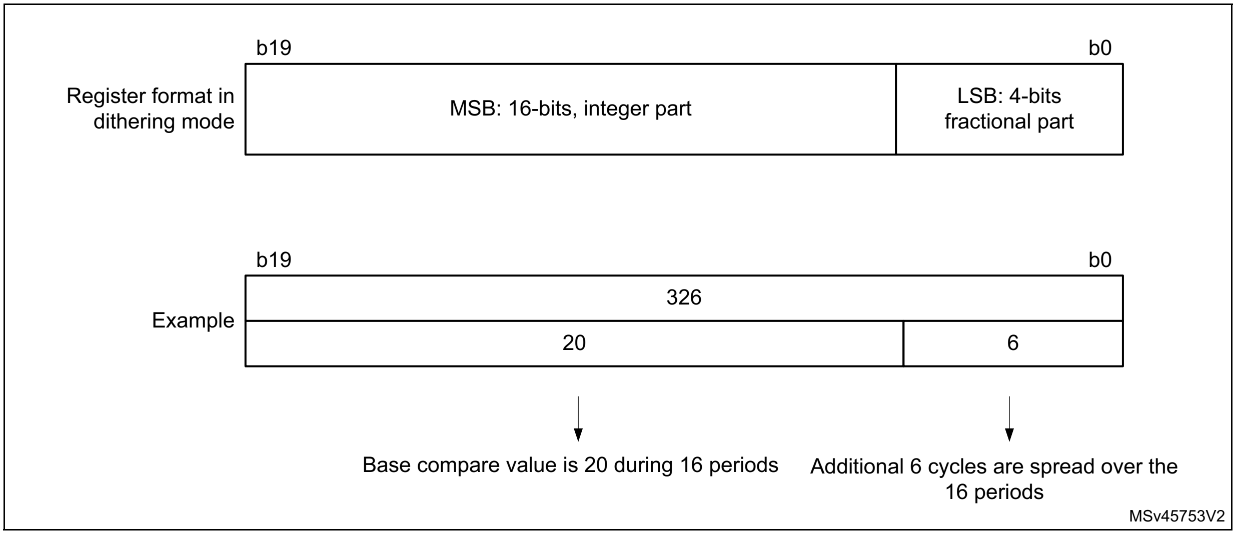


When the dithering mode is enabled, the register coding is changed as following (see *Figure 461* for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- the MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: *The ARR and CCR values will be updated automatically if the DITHEN bit is set / reset (for instance, if ARR= 0x05 with DITHEN=0, it will be updated to ARR = 0x50 with DITHEN = 1).*
The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The DITHEN bit must be reset
4. The CCIF flags must be cleared
5. The CEN bit can be set (eventually with ARPE = 1).

Figure 461. Data format and register coding in dithering mode


b19
b0
Register format in dithering mode
MSB: 16-bits, integer part
LSB: 4-bits fractional part
b19
b0
Example
326
20
6
Base compare value is 20 during 16 periods
Additional 6 cycles are spread over the 16 periods
MSv45753V2

The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \Rightarrow F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{\text{Max}_{\text{Resolution}}}$$

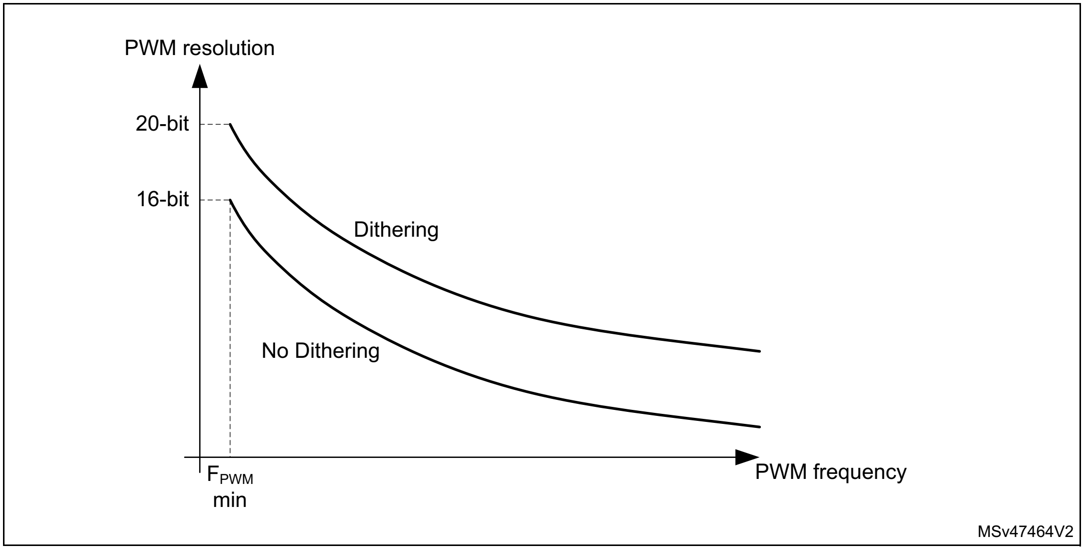
$$\text{Dithering mode disabled: } F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65536}$$

$$\text{Dithering mode enabled: } F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

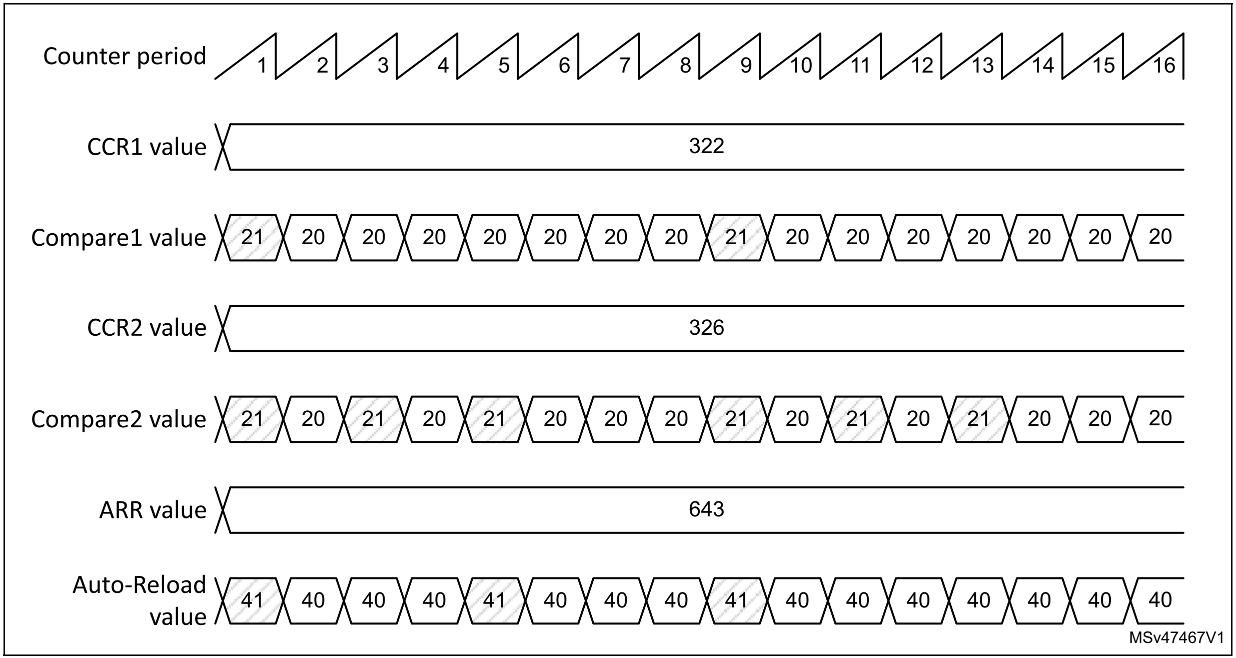
As shown on the *Figure 462* below, the dithering mode allows to increase the PWM resolution whatever the PWM frequency.

Figure 462. PWM resolution vs frequency



The duty cycle and / or period changes are spread over 16 consecutive periods, as described in the *Figure 463* below.

Figure 463. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 294* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 294. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 294. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

30.4.13 Combined PWM mode (TIM15 only)

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, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one tim_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.

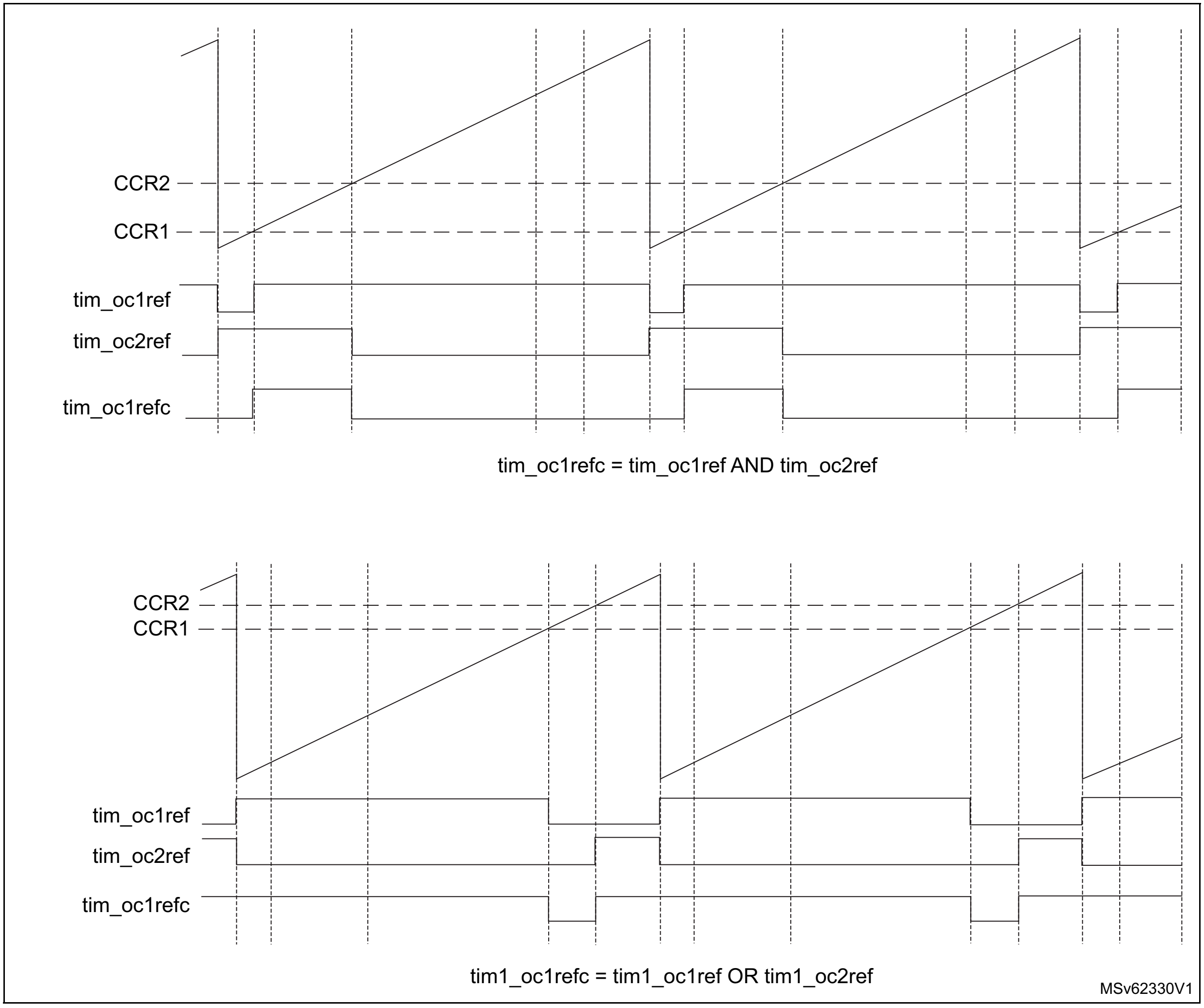
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 464 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,

Figure 464. Combined PWM mode on channel 1 and 2



30.4.14 Complementary outputs and dead-time insertion

The TIM15/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 tim_ocx or complementary tim_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 tim_ocx and tim_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 301: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 1433* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

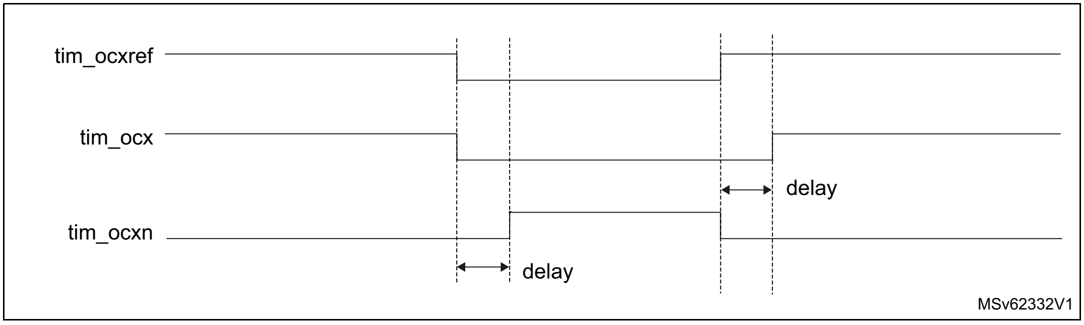
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates 2 outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_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 tim_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 (tim_ocx or tim_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 tim_ocxref. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 465. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIMx_DTR2 allows to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 466*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must be not modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 466. Asymmetrical deadtime

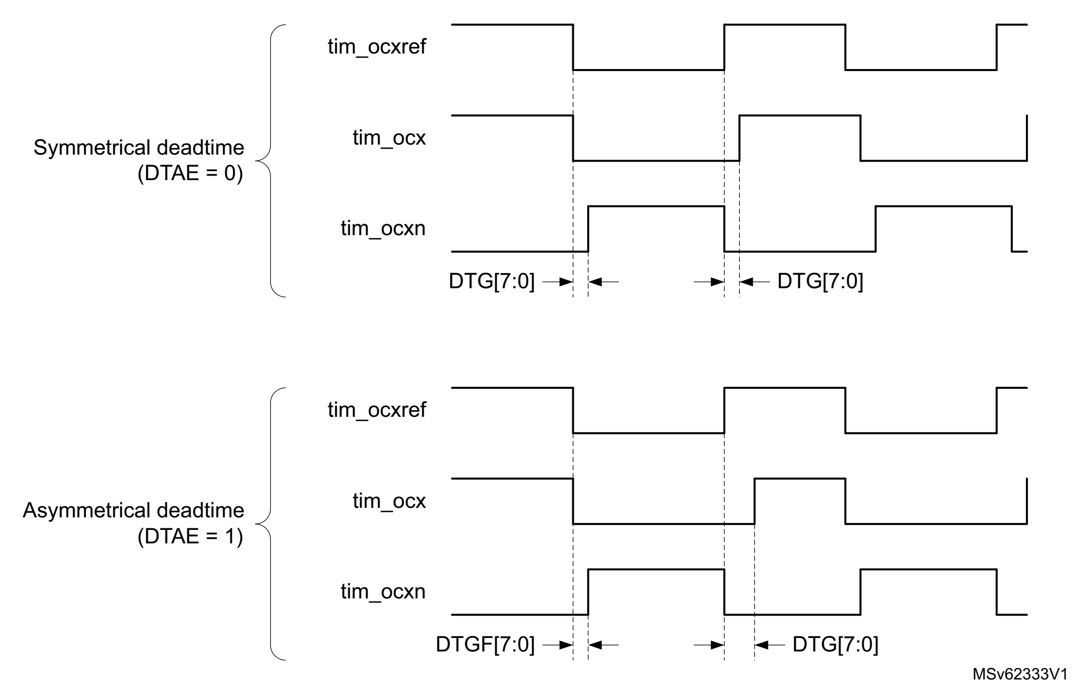


Figure 467. Dead-time waveforms with delay greater than the negative pulse.

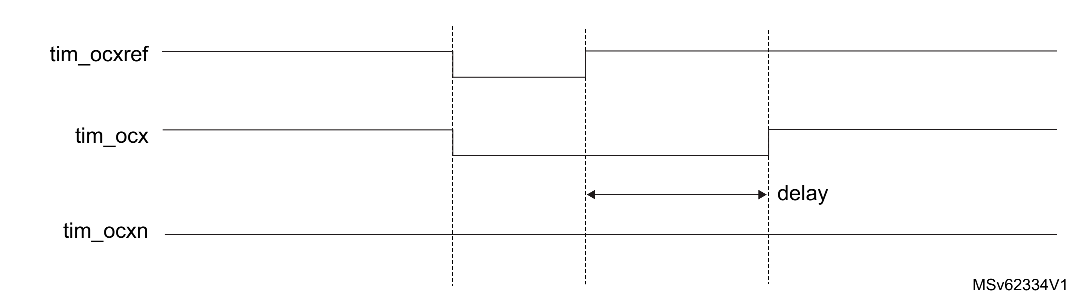
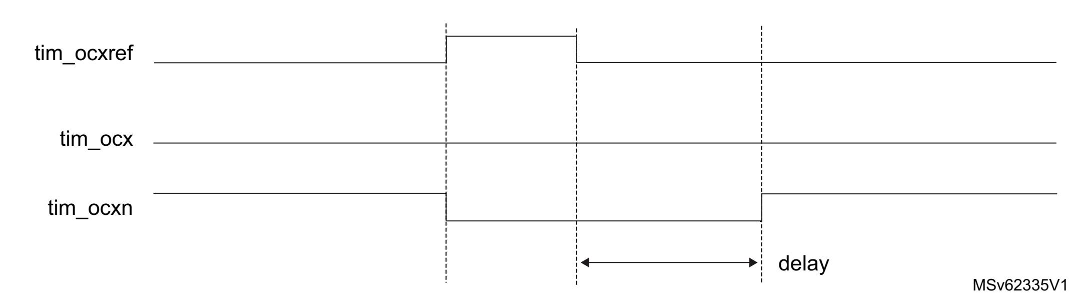


Figure 468. Dead-time waveforms with delay greater than the positive pulse.



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 30.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 1437* for delay calculation.

Re-directing tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be re-directed to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows to send a specific waveform (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 tim_ocxn is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP=0 then tim_ocxn=tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE=CCxNE=1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

30.4.15 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the 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.

The break channel gathers both system-level fault (clock failure, ECC / parity errors,...) 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.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows to enable /disable the outputs 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 tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 299: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) on page 1407* for more details.

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.

The break is generated by the tim_brk inputs which has:

- Programmable polarity (BKP bit in the TIMx_BDTR register)
- Programmable enable bit (BKE bit in the TIMx_BDTR register)
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

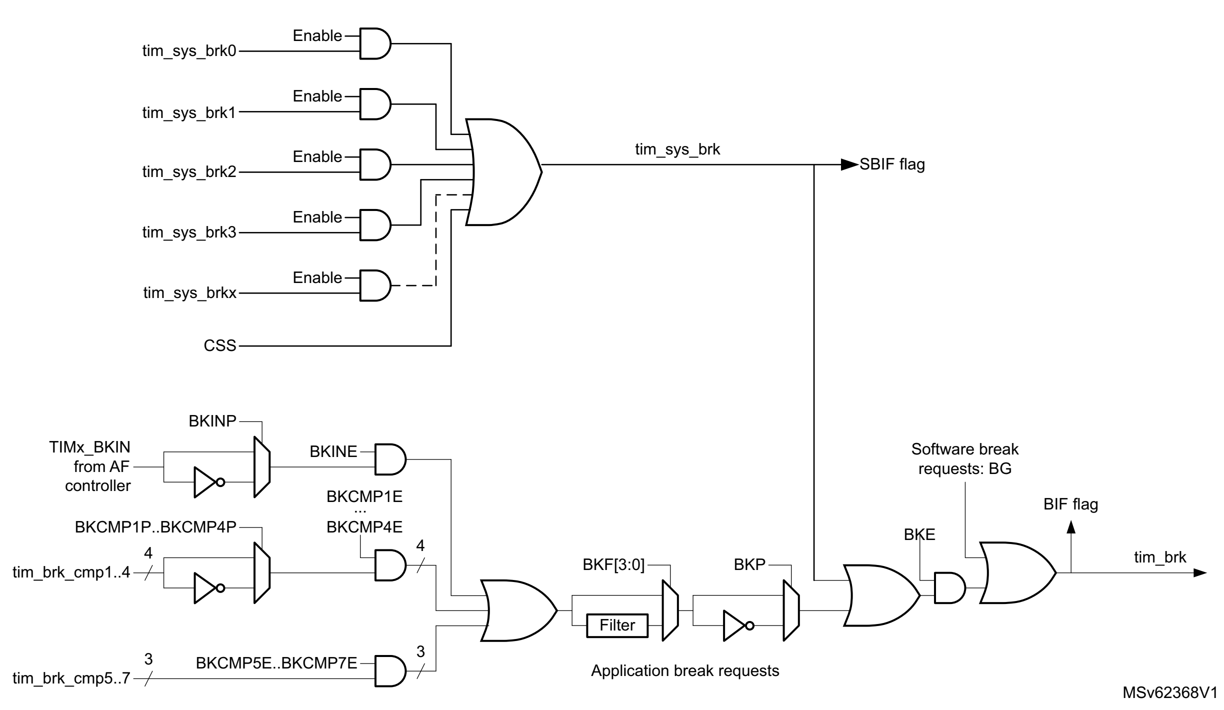
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  – coming from a tim_brk_cmpx input (refer to *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation)
  – coming from a system break request on the tim_sys_brk inputs (refer to *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 469* below.

Figure 469. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

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, tim_ocx and tim_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 tim_ker_ck 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. A DMA request can be sent if the BDE 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.

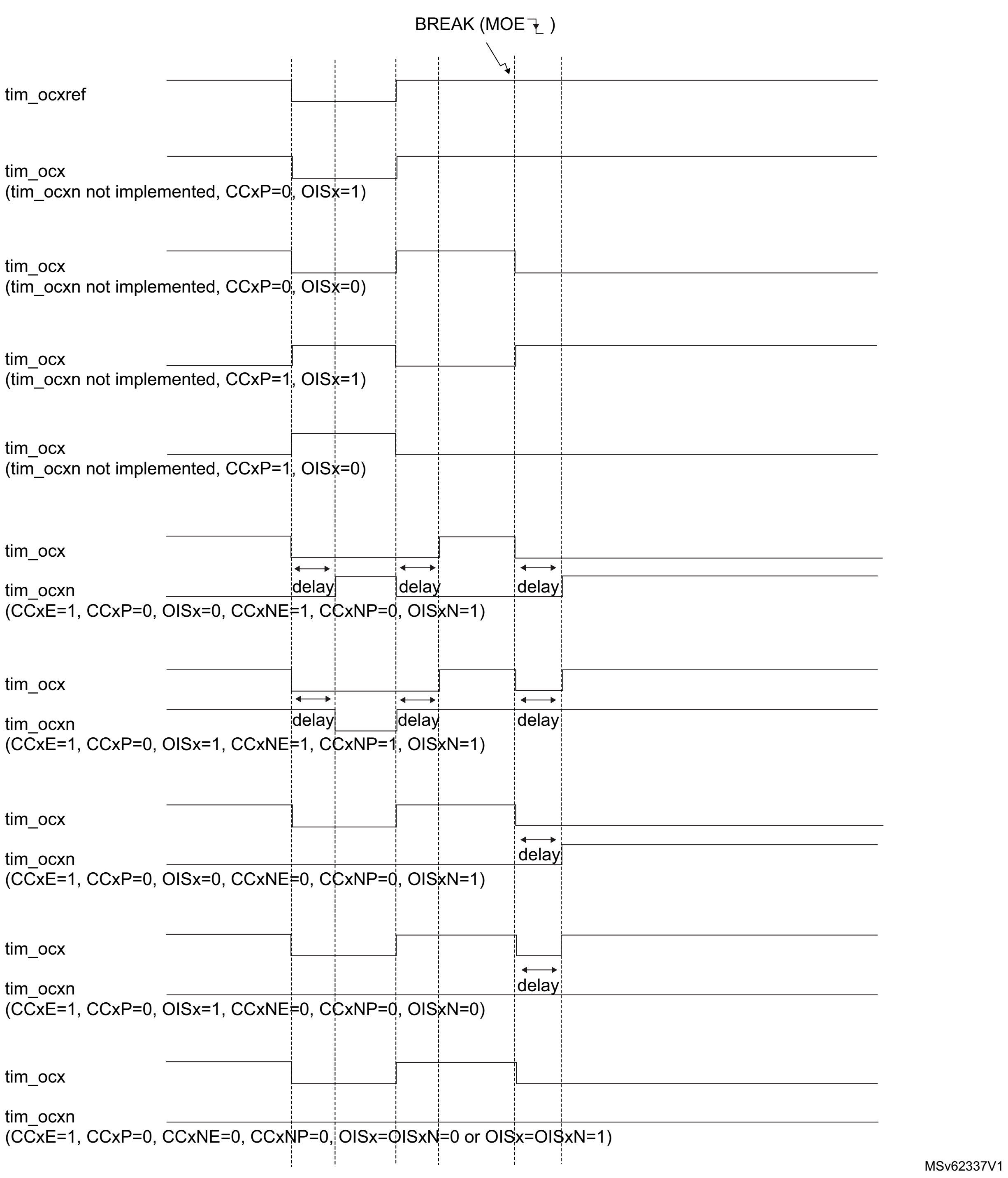
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_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 to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_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 30.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 1437*. The LOCK bits can be written only once after an MCU reset.

The *Figure 470* shows an example of behavior of the outputs in response to a break.

Figure 470. Output behavior in response to a break event on tim_brk


BREAK (MOE ↓)
tim_ocxref
tim_ocx
(tim_ocxn not implemented, CCxP=0, OISx=1)
tim_ocx
(tim_ocxn not implemented, CCxP=0, OISx=0)
tim_ocx
(tim_ocxn not implemented, CCxP=1, OISx=1)
tim_ocx
(tim_ocxn not implemented, CCxP=1, OISx=0)
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, OISx=0, CCxNE=1, CCxNP=0, OISxN=1)
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=1, OISxN=1)
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISxN=1)
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, OISx=1, CCxNE=0, CCxNP=0, OISxN=0)
tim_ocx
tim_ocxn
(CCxE=1, CCxP=0, CCxNE=0, CCxNP=0, OISx=OISxN=0 or OISx=OISxN=1)
delay
MSv62337V1

30.4.16 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 471*.

They allow to have:

- 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 sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The tim_brk 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 (triggered by setting the BG bit) 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 TIM_BKIN 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 295*)

Table 295. Break protection disarming conditions

MOE	BKBID	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

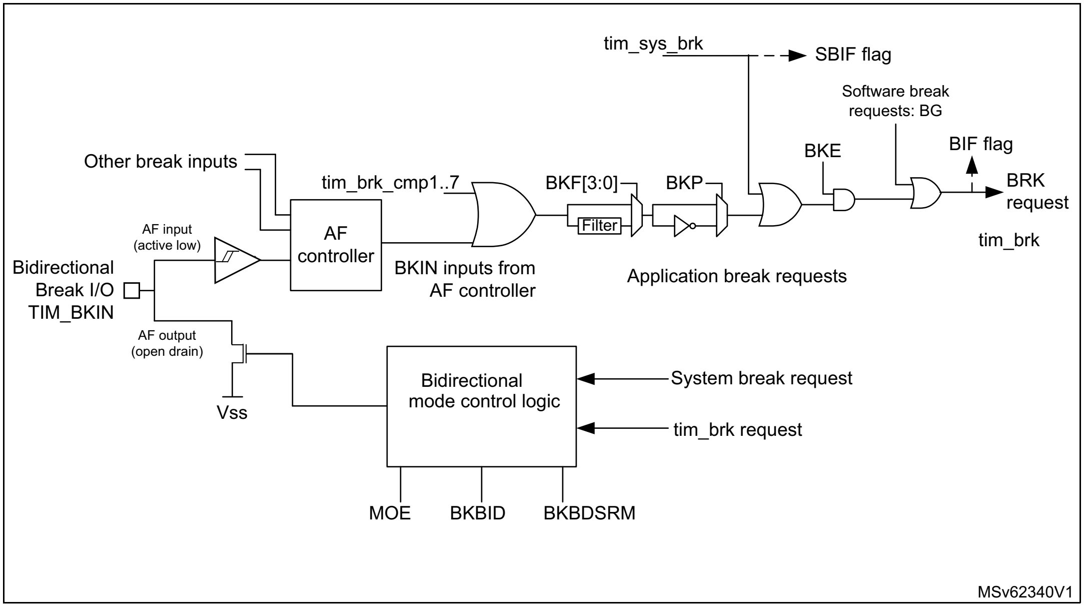
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

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 471. Output redirection

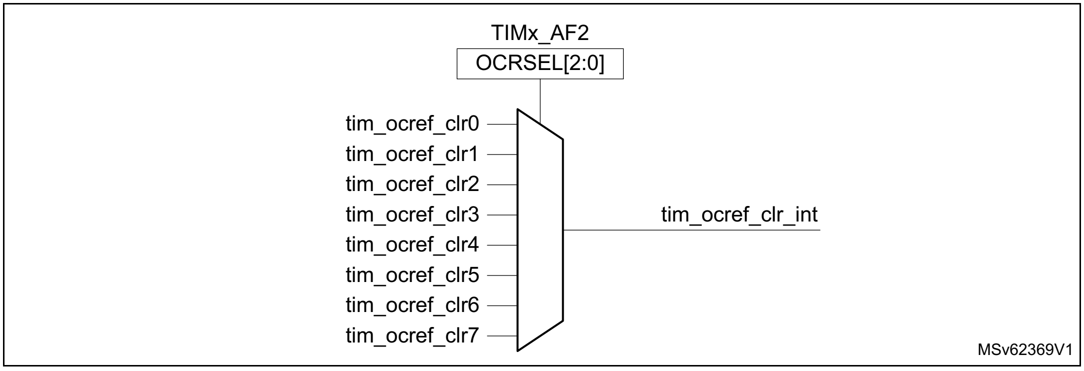


30.4.17 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_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.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 472* below.

Figure 472. tim_ocref_clr input selection multiplexer



30.4.18 One-pulse mode

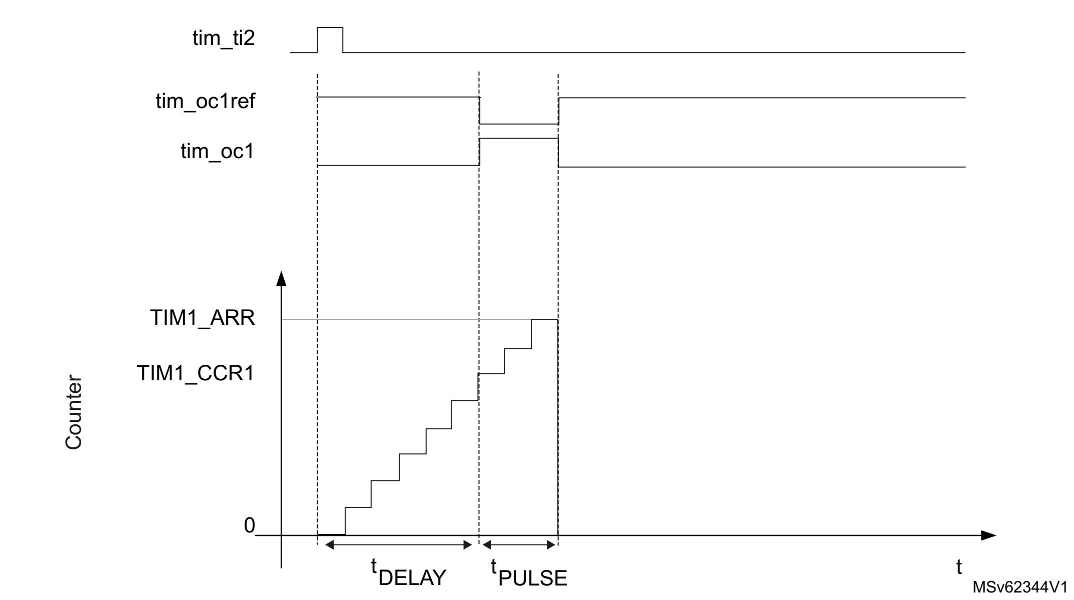
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 473. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let’s use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[1..15] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S=’01’ in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P=’0’ and CC2NP=’0’ in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS=’00110’ in the TIMx_SMCR register.
5. tim_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 tim_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: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_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 tim_ocxref (and tim_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.

30.4.19 Retriggerable one pulse mode (TIM15 only)

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 30.4.18*:

- – 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

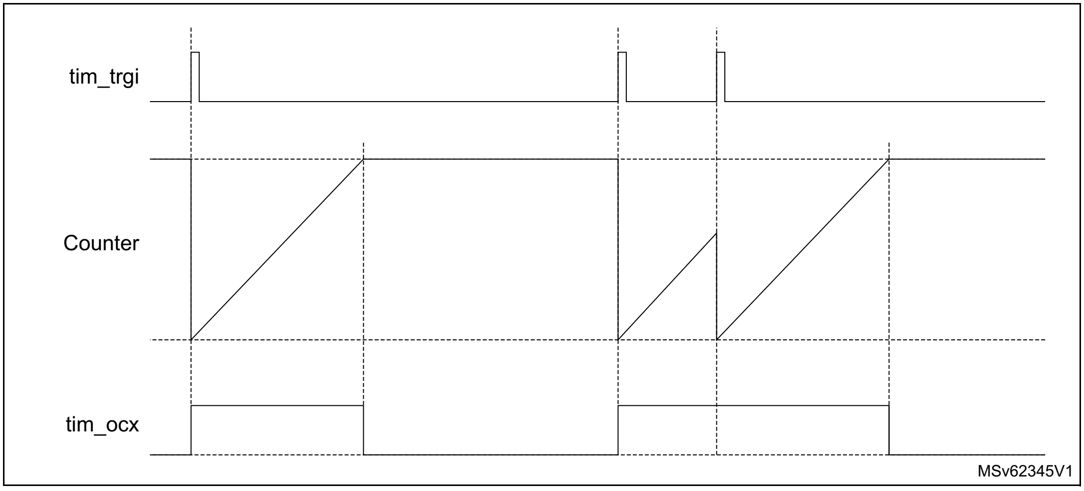
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retrigerrable OPM mode 1 or 2.

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 474. Retriggerable one pulse mode



30.4.20 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 to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. 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).

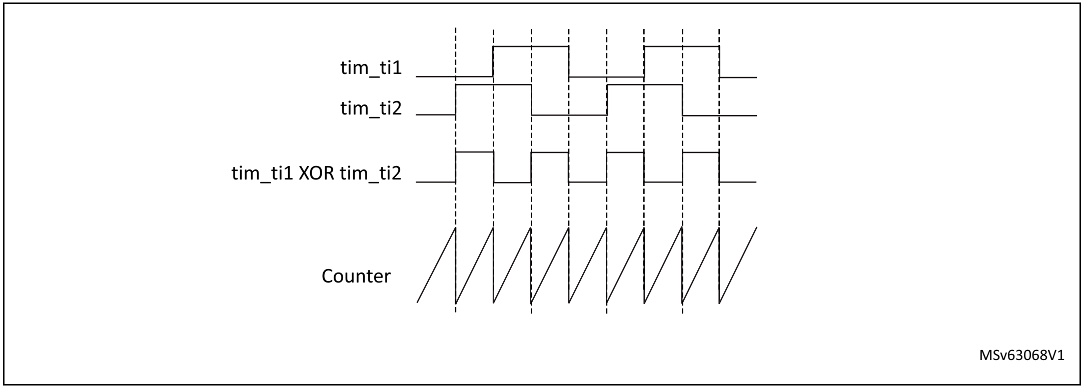
There is no latency between the assertions of the UIF and UIFCPY flags.

30.4.21 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 475*.

Figure 475. Measuring time interval between edges on 2 signals



30.4.22 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

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.

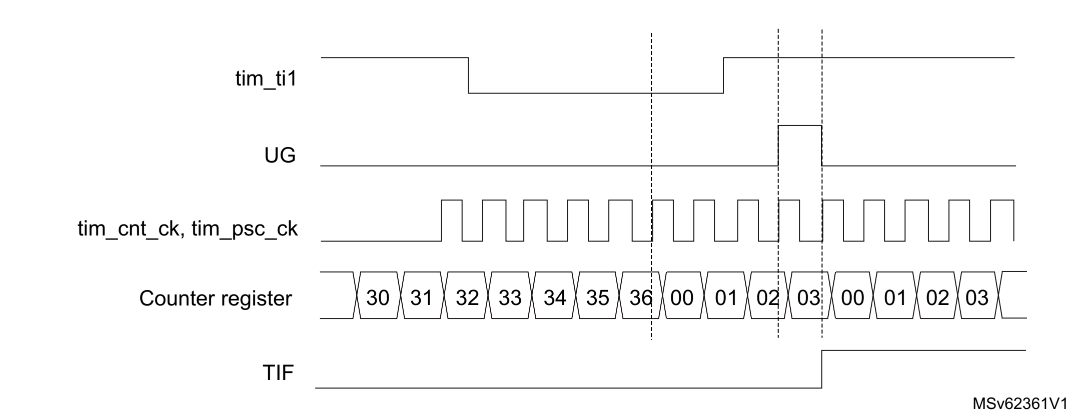
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_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 the 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 tim_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 tim_ti1 rising edge. When tim_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 tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 476. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

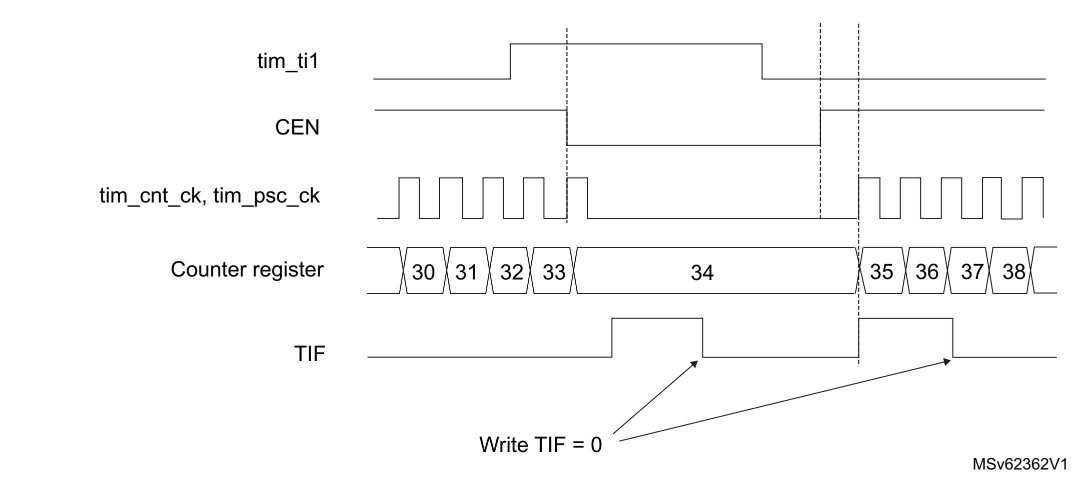
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_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 the 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 tim_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 tim_ti1 is low and stops as soon as tim_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 tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 477. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

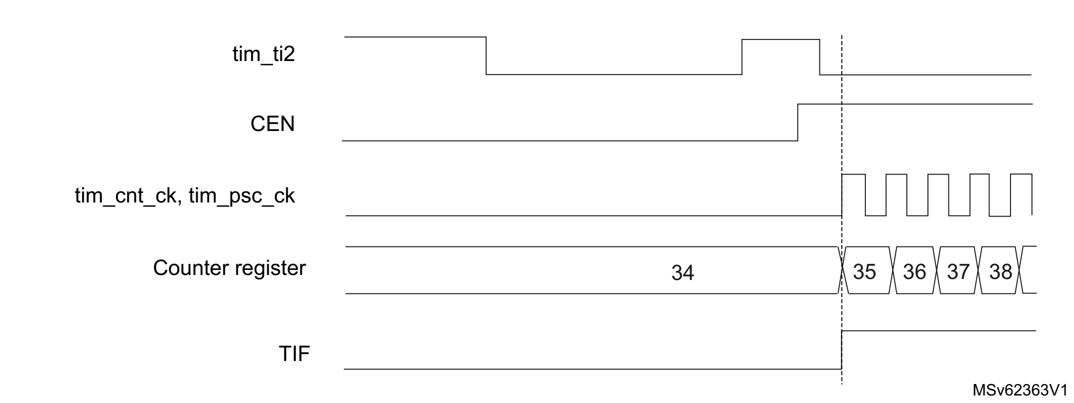
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_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 the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in the TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 478. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

30.4.23 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

30.4.24 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode allows to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

30.4.25 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 22.3.19: Timer synchronization* for details.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_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.

30.4.26 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 timer 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 detects 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.

30.4.27 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.

30.4.28 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA requests, as shown in *Table 296*.

Table 296. DMA request

DMA acronym	DMA request	Enable control bit
TIM_UP	Update	UDE
TIM_CH1	Capture/compare 1	CC1DE
TIM_COM	Commutation (COM)	COMDE
TIM_TRIG(1)	Trigger	TDE

1. Available for TIM15 only.

30.4.29 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) 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) to force them to Hi-Z.

For more details, refer to the debug section.

30.5 TIM15/TIM16/TIM17 low-power modes

Table 297. Effect of low-power modes on TIM15/TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

30.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in *Table 298*.

Table 298. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No
	Break	BIF	BIE	write 0 in BIF	Yes	No

1. Available for TIM15 only.

30.7 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

30.7.1 TIM15 control register 1 (TIM15_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.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM15_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM15_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)

00: $t_{DTS} = t_{tim\_ker\_ck}$
01: $t_{DTS} = 2^{*}t_{tim\_ker\_ck}$
10: $t_{DTS} = 4^{*}t_{tim\_ker\_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM15_ARR register is not buffered
1: TIM15_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 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 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.

30.7.2 TIM15 control register 2 (TIM15_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.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)

0: tim_oc2=0 when MOE=0

1: tim_oc2=1 when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n=0 after a dead-time when MOE=0

1: tim_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 TIM15_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1=0 after a dead-time when MOE=0

1: tim_oc1=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 TIM15_BKR register).

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15..0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15..0] and tim_ti2_in[15..0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: **Reset** - the UG bit from the TIM15_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_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 tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM15_SMCR register).

010: **Update** - The update event is selected as trigger output (tim_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 (tim_trgo).

100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

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 tim_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 tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

30.7.3 TIM15 slave mode control register (TIM15_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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								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.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_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 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
Others: Reserved

See *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for more details on tim_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 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_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: Reserved

0010: Reserved

0011: Reserved

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_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 tim_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 (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=’00100’). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_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.

30.7.4 TIM15 DMA/interrupt enable register (TIM15_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	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	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

Bits 12:10 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 **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

Bits 4:3 Reserved, must be kept at reset value.

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

30.7.5 TIM15 status register (TIM15_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.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

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 TIM15_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 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). 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 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 4:3 Reserved, must be kept at reset value.

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 downcounting 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 TIM15_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIM15_EGR register, if URS=0 and UDIS=0 in the TIM15_CR1 register.
- When CNT is reinitialized by a trigger event (refer to *Section 30.7.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS=0 and UDIS=0 in the TIM15_CR1 register.

30.7.6 TIM15 event generation register (TIM15_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.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			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 **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 TIM15_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 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 4:3 Reserved, must be kept at reset value.

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 TIM15_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).

30.7.7 TIM15 capture/compare mode register 1 [alternate] (TIM15_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).

31	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

Input capture mode

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, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = ‘0’ in TIM15_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_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: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, 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 (tim_ic1). The prescaler is reset as soon as CC1E='0' (TIM15_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, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIM15_CCER).

30.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_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).

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

Output compare mode:

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

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, tim_ic2 is mapped on tim_ti2.
10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = ‘0’ in TIM15_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input.
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIM15_CCR1 and the counter TIM15_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM15_CNT=TIM15_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIM15_CNT<TIM15_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIM15_CNT<TIM15_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_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 tim_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 tim_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 tim_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 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved,

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode, the tim_ocxref level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0:Preload register on TIM15_CCR1 disabled. TIM15_CCR1 can be written at anytime, the new value is taken in account immediately.

1:Preload register on TIM15_CCR1 enabled. Read/Write operations access the preload register. TIM15_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 TIM15_BDTR register) and CC1S='00' (the channel is configured in output).

The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIM15_CR1 register). Else the behavior is not guaranteed.

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, tim_ocx 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. 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, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIM15_CCER).

30.7.9 TIM15 capture/compare enable register (TIM15_CCER)

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.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary 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 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_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 TIM15_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 TIM15_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 - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_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

CC1 channel configured as output:

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 TIM15_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_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 299* for details.

Table 299. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_ocx=0 tim_ocxn=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_ocx=0	tim_ocxref + Polarity tim_ocxn=tim_ocxref XOR CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn=0
		X	1	1	tim_ocxref + Polarity + dead-time	Complementary to tim_ocxref (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx=CCxP	tim_ocxref + Polarity tim_ocxn=tim_ocxref XOR CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_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: tim_ocx=CCxP, tim_ocxn=CCxNP Then if the clock is present: tim_ocx=OISx and tim_ocxn=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_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 tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

30.7.10 TIM15 counter (TIM15_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 in the TIM15_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

30.7.11 TIM15 prescaler (TIM15_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 ($f_{tim\_cnt\_ck}$) is equal to $f_{tim\_psc\_ck}$ / (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 TIM15_EGR register or through trigger controller when configured in “reset mode”).

30.7.12 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	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.	ARR[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 30.4.3: Time-base unit on page 1353* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

30.7.13 TIM15 repetition counter register (TIM15_RCR)

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 reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the reptition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM15_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 PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode

30.7.14 TIM15 capture/compare register 1 (TIM15_CCR1)

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.	CCR1[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: 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 TIM15_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 TIM15_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

30.7.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

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.	CCR2[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: 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 TIM15_CCMR2 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 TIM15_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

30.7.16 TIM15 break and dead-time register (TIM15_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.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			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 BKBID, BKDSRM, BKF[3:0], 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 TIM15_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_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 TIM15_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 tim_brk is armed
1: Break input tim_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 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 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, 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 TIM15_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk 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: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM15_CCER register)

See tim_ocx/tim_ocxn enable description for more details (*Section 30.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1405*).

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 TIM15_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_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 (tim_brk and tim_sys_brk clock failure event) disabled
1; Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_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 that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (*Section 30.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1405*).

0: When inactive, tim_ocx/tim_ocxn 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, tim_ocx/tim_ocxn 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 TIM15_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (*Section 30.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1405*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal=0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. tim_ocx/tim_ocxn enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_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 TIM15_BDTR register, OISx and OISxN bits in TIM15_CR2 register and BKBID/BKE/BKP/AOE bits in TIM15_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM15_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: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM15_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 TIM15_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])xt_{dtg} with T_{dtg}=2xt_{DTS}

DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with T_{dtg}=8xt_{DTS}

DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with T_{dtg}=16xt_{DTS}

Example if T_{DTS}=125ns (8MHz), 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 TIM15_BDTR register).

30.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2)

Address offset: 0x054

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.	DTPE	DTAE
														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.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x t_{dtg} with t_{dtg}=t_{DTS}.
DTGF[7:5]=10x => DTF=(64+DTGF[5:0])xt_{dtg} with T_{dtg}=2xt_{DTS}.
DTGF[7:5]=110 => DTF=(32+DTGF[4:0])xt_{dtg} with T_{dtg}=8xt_{DTS}.
DTGF[7:5]=111 => DTF=(32+DTGF[4:0])xt_{dtg} with T_{dtg}=16xt_{DTS}.
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

30.7.18 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x5C

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]**: selects tim_ti2_in[0..15] input

0000: TIM15_CH2 input (tim_ti2_in0)
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[0..15] input

0000: TIM15_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

30.7.19 TIM15 alternate function register 1 (TIM15_AF1)

Address offset: 0x060

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.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. mdf_brkx output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. COMP7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled
1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled
1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled
1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

30.7.20 TIM15 alternate function register 2 (TIM15_AF2)

Address offset: 0x064

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.	OCRSEL[2:0]		
													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 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0
001: tim_ocref_clr1
010: tim_ocref_clr2
011: tim_ocref_clr3
100: tim_ocref_clr4
101: tim_ocref_clr5
110: tim_ocref_clr6
111: tim_ocref_clr7

Refer to *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

30.7.21 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x3DC

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.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 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 TIM15_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 field defines the base-address for DMA transfers (when read/write access are done through the TIM15_DMAR address). DBA is defined as an offset starting from the address of the TIM15_CR1 register.

Example:

00000: TIM15_CR1,
00001: TIM15_CR2,
00010: TIM15_SMCR,
...

30.7.22 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x3E0

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

(TIM15_CR1 address) + (DBA + DMA index) x 4

where TIM15_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM15_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM15_DCR).

30.7.23 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 300. TIM15 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	**TIM15_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	**TIM15_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value																						0	0	0	0	0	0	0	0	0		0
0x08	**TIM15_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x0C	**TIM15_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0				0	0	0	0	0			0	0	0
0x10	**TIM15_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0
0x14	**TIM15_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	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	**TIM15_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
	TIM15_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

Table 300. TIM15 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
0x20	TIM15_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.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	TIM15_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	TIM15_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	TIM15_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIM15_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	TIM15_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIM15_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	TIM15_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	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
0x48 - 0x50	Reserved	Res.																															
0x54	TIM15_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	TIM15_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

Table 300. TIM15 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
0x60	TIM15_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	TIM15_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68 - 0x3D8	Reserved	Res.																															
0x3DC	TIM15_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
0x3E0	TIM15_DMAR	DMAB[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 2.2 on page 81* for the register boundary addresses.

30.8 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

30.8.1 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.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

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 (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),

00: t_{DTS}=$t_{tim\_ker\_ck}$

01: t_{DTS}=2*$t_{tim\_ker\_ck}$

10: t_{DTS}=4*$t_{tim\_ker\_ck}$

11: Reserved

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: nly 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.

30.8.2 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 (tim_oc1n output)

0: tim_oc1n=0 after a dead-time when MOE=0
1: tim_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_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1=0 after a dead-time when MOE=0
1: tim_oc1=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_BKR 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 a rising edge occurs on tim_trgi (if available).

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.

30.8.3 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.	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
		rw				rw	rw	rw		rw				rw	rw

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:10 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

30.8.4 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 tim_brk 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.

30.8.5 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).

30.8.6 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 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).

31	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

Input capture mode

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 tim_ti1 input and the length of the digital filter applied to tim_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: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, 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 (tim_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, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = ‘0’ in TIMx_CCER).

30.8.7 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 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).

31	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.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

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

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_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. tim_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. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_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.

Others: 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 tim_oc1ref level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

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).

The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

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, tim_ocx 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, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

30.8.8 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: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. 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 - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_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 301* for details.

Table 301. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CC1E bit	CC1NE bit	tim_oc1 output state	tim_oc1n output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_oc1=0 tim_oc1n=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_oc1=0	tim_oc1ref + Polarity tim_oc1n=tim_oc1ref XOR CC1NP
		0	1	0	tim_oc1ref + Polarity tim_oc1=tim_oc1ref XOR CC1P	Output Disabled (not driven by the timer: Hi-Z) tim_oc1n=0
		X	1	1	tim_oc1ref + Polarity + dead-time	Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_oc1=CC1P	tim_oc1ref + Polarity tim_oc1n=tim_oc1ref XOR CC1NP
		1	1	0	tim_oc1ref + Polarity tim_oc1=tim_oc1ref XOR CC1P	Off-State (output enabled with inactive state) tim_oc1n=CC1NP
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: tim_oc1=CC1P, tim_oc1n=CC1NP Then if the clock is present: tim_oc1=OIS1 and tim_oc1n=OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n 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 OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

30.8.9 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.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

30.8.10 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 (tim_cnt_ck) is equal to $f_{tim\_psc\_ck}$ / (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”).

30.8.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	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.	ARR[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 30.4.3: Time-base unit on page 1353* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

30.8.12 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 reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.
When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, 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 PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode

30.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

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.	CCR1[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: 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 tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

30.8.14 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.	BKF[3:0]			
			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 BKBID, BKDSRM, BKF[3:0], 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 tim_brk in input mode

1: Break input tim_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 tim_brk is armed

1: Break input tim_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 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING}$=$f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}$=$f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}$=$f_{tim\_ker\_ck}$, 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

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 the tim_brk 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: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 30.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1431*).

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 tim_brk input is not 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 tim_brk is active low
1: Break input tim_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 (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk 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 tim_oc1/tim_oc1n enable description for more details (*Section 30.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1431*).

0: When inactive, tim_oc1/tim_oc1n 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, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E=1 or CC1NE=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 tim_oc1/tim_oc1n enable description for more details (*Section 30.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1431*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal=0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E=1 or CC1NE=1. tim_oc1/tim_oc1n 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 BKBID/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])xt_{dtg} with T_{dtg}=2xt_{DTS}
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with T_{dtg}=8xt_{DTS}
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with T_{dtg}=16xt_{DTS}
Example if T_{DTS}=125ns (8MHz), 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).

30.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17)

Address offset: 0x054

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.	DTPE	DTAE
														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.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

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 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

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 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x t_{dtg} with t_{dtg}=t_{DTS}.
DTGF[7:5]=10x => DTF=(64+DTGF[5:0])xt_{dtg} with T_{dtg}=2xt_{DTS}.
DTGF[7:5]=110 => DTF=(32+DTGF[4:0])xt_{dtg} with T_{dtg}=8xt_{DTS}.
DTGF[7:5]=111 => DTF=(32+DTGF[4:0])xt_{dtg} with T_{dtg}=16xt_{DTS}.
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us 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).

30.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17)

Address offset: 0x5C

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 tim_ti1_in[0..15] input

0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

30.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)

Address offset: 0x060

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.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high
1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high
1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high
1: tim_brk_cmp2 input is active low

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**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high
1: tim_brk_cmp1 input is active low

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**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high
1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. mdf_brkx output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled
1: tim_brk_cmp8 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 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled
1: tim_brk_cmp7 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 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled
1: tim_brk_cmp6 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 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled
1: tim_brk_cmp5 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 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled
1: tim_brk_cmp4 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 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled
1: tim_brk_cmp3 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 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled
1: tim_brk_cmp2 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**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled
1: tim_brk_cmp1 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**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled
1: TIMx_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).

30.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17)

Address offset: 0x064

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.	OCRSEL[2:0]		
													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 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection

These bits select the tim_ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
010: tim_ocref_clr2
011: tim_ocref_clr3
100: tim_ocref_clr4
101: tim_ocref_clr5
110: tim_ocref_clr6
111: tim_ocref_clr7
Refer to *Section 30.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

30.8.19 TIMx option register 1 (TIMx_OR1)(x = 16 to 17)

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.	Res.	Res.	Res.	HSE32 EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **HSE32EN**: HSE Divided by 32 enable

This bit enables the HSE divider by 32 for the tim_ti1_in3. See *Table 288: Interconnect to the tim_ti1 input multiplexer* for details.
0: HSE divided by 32 disabled
1: HSE divided by 32 enabled

30.8.20 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x3DC

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.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 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.

30.8.21 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x3E0

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).

30.8.22 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 302. 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.	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																			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.	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 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]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	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.	OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value																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.	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.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 302. 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.	CCR1[19:0]																			
	Reset value													0	0	0	0	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.	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
0x48 - 0x50	Reserved	Res.																															
0x54	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	TIMx_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
0x60	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68	TIMx_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.	Res.	HSE32EN
	Reset value																																0
0x6C - 0x3D8	Reserved	Res.																															
0x3DC	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
0x3E0	TIMx_DMAR	DMAB[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 2.2* for the register boundary addresses.

31 Basic timers (TIM6/TIM7)

31.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist in a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

31.2 TIM6/TIM7 main features

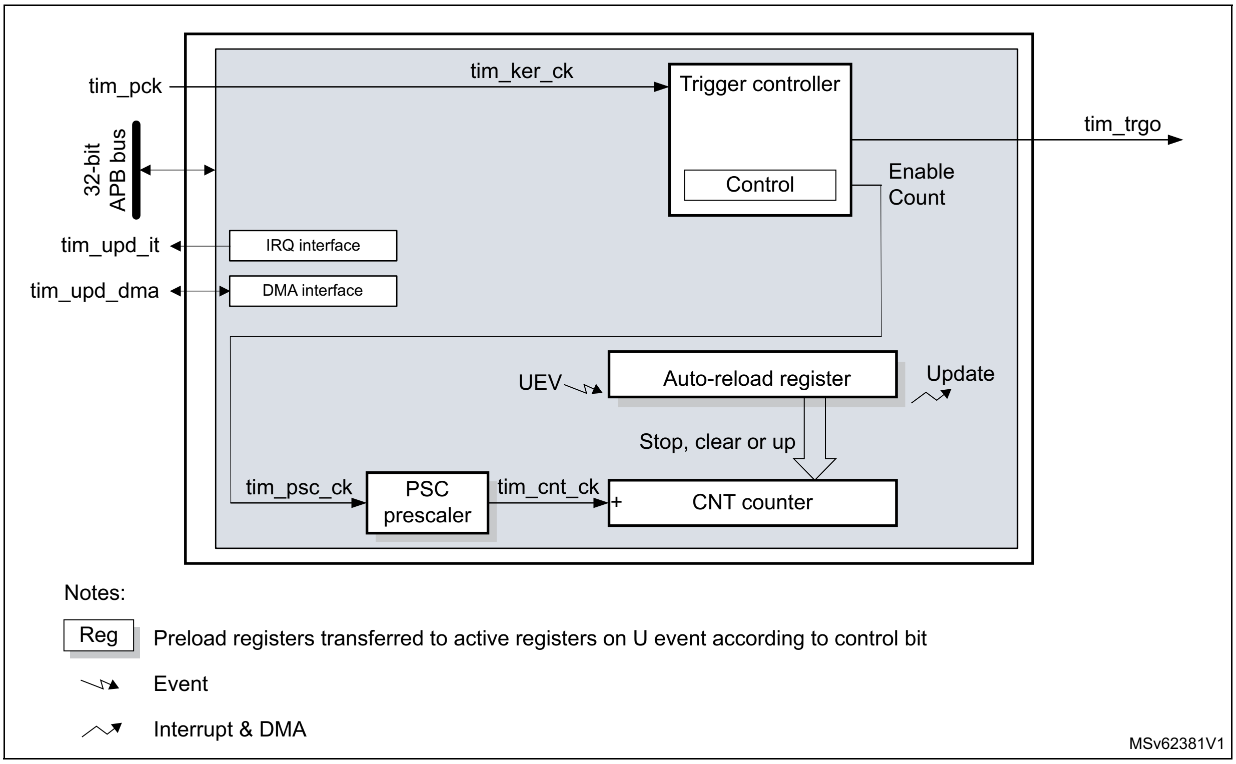
Basic timer (TIM6/TIM7) features include:

- 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
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

31.3 TIM6/TIM7 functional description

31.3.1 TIM6/TIM7 block diagram

Figure 479. Basic timer block diagram



31.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 303. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

31.3.3 TIM6/TIM7 clocks

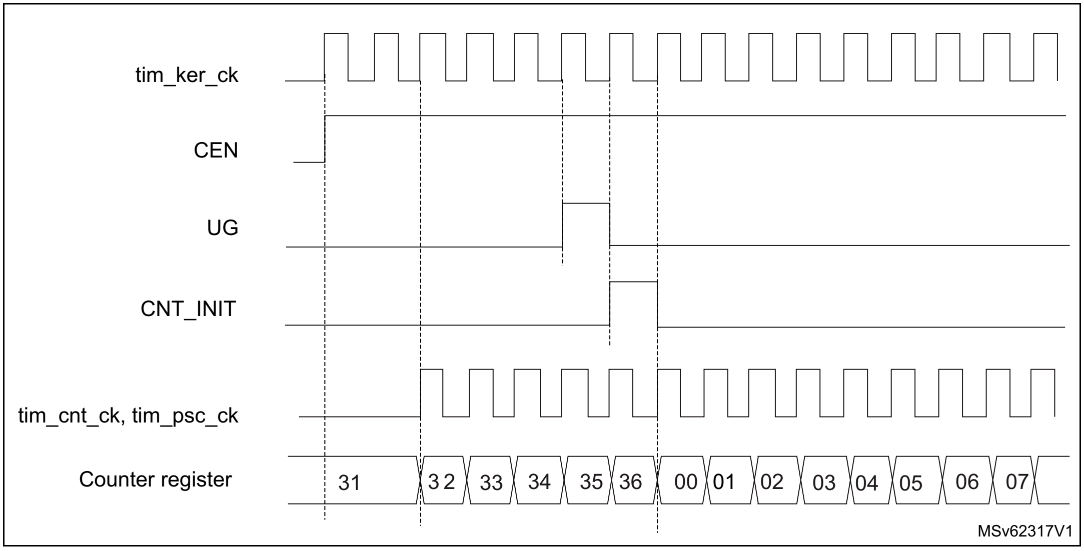
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

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 for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 480 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 480. Control circuit in normal mode, internal clock divided by 1



31.3.4 Time-base unit

The main block of the programmable 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)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents 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 the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value 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 tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set 1 clock cycle after CEN bit set.

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 the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 481 and *Figure 482* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 481. Counter timing diagram with prescaler division change from 1 to 2

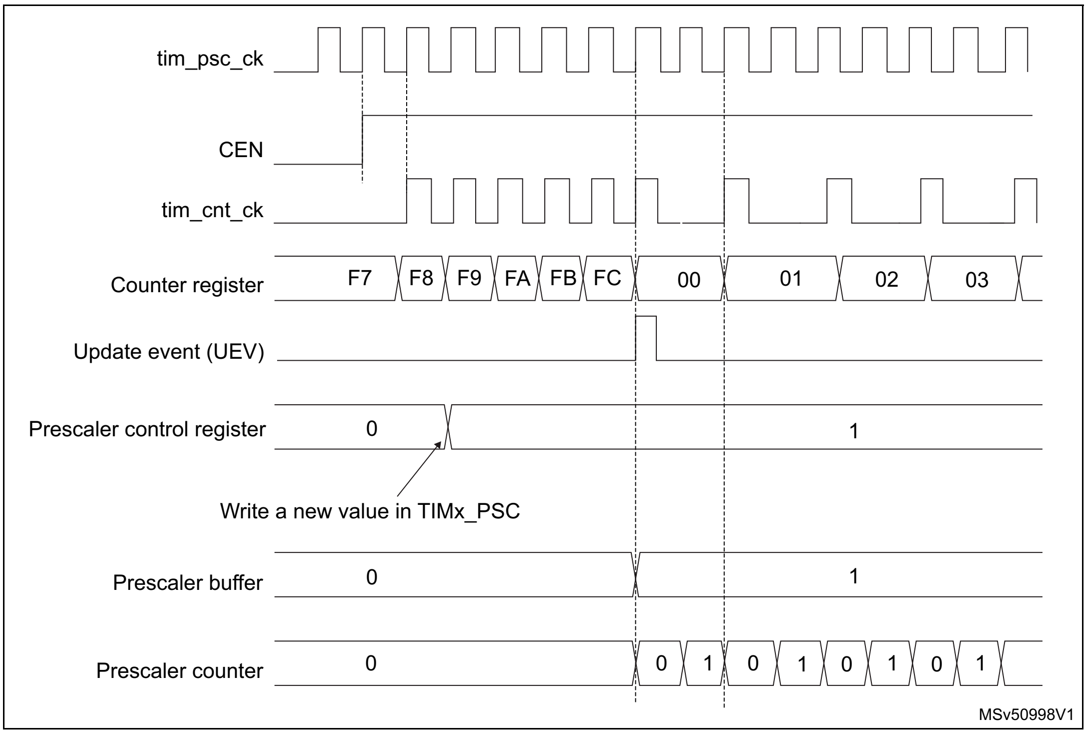
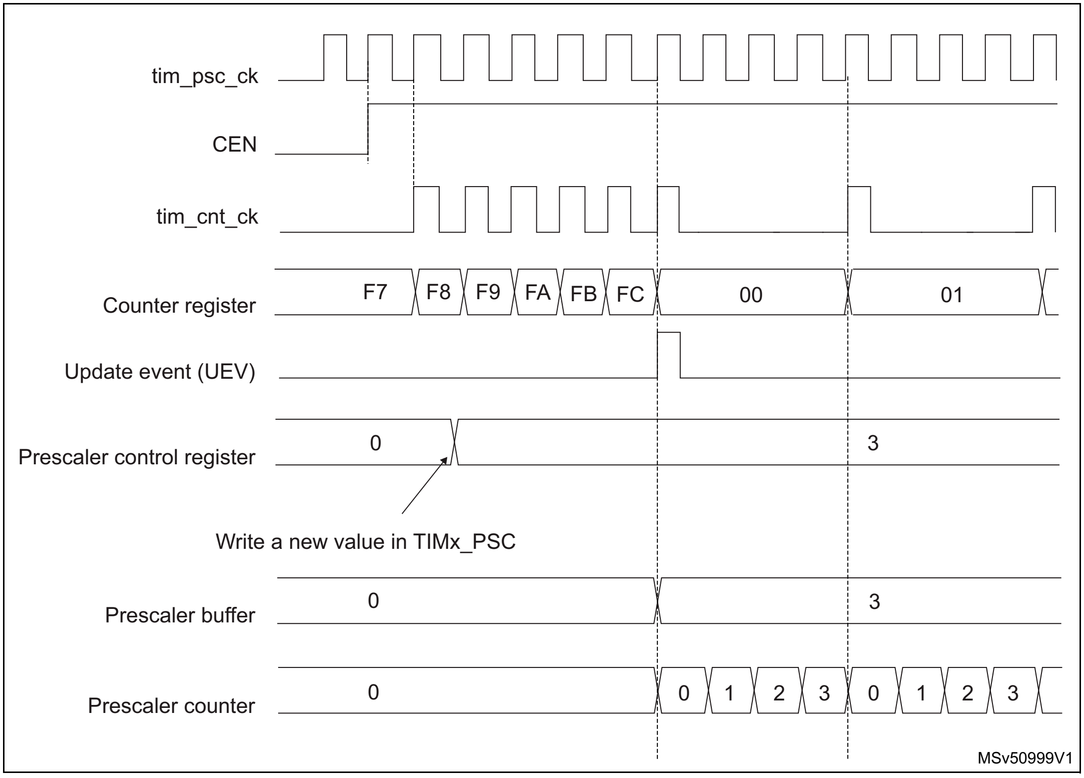


Figure 482. Counter timing diagram with prescaler division change from 1 to 4



31.3.5 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate 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 the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents 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 483. Counter timing diagram, internal clock divided by 1

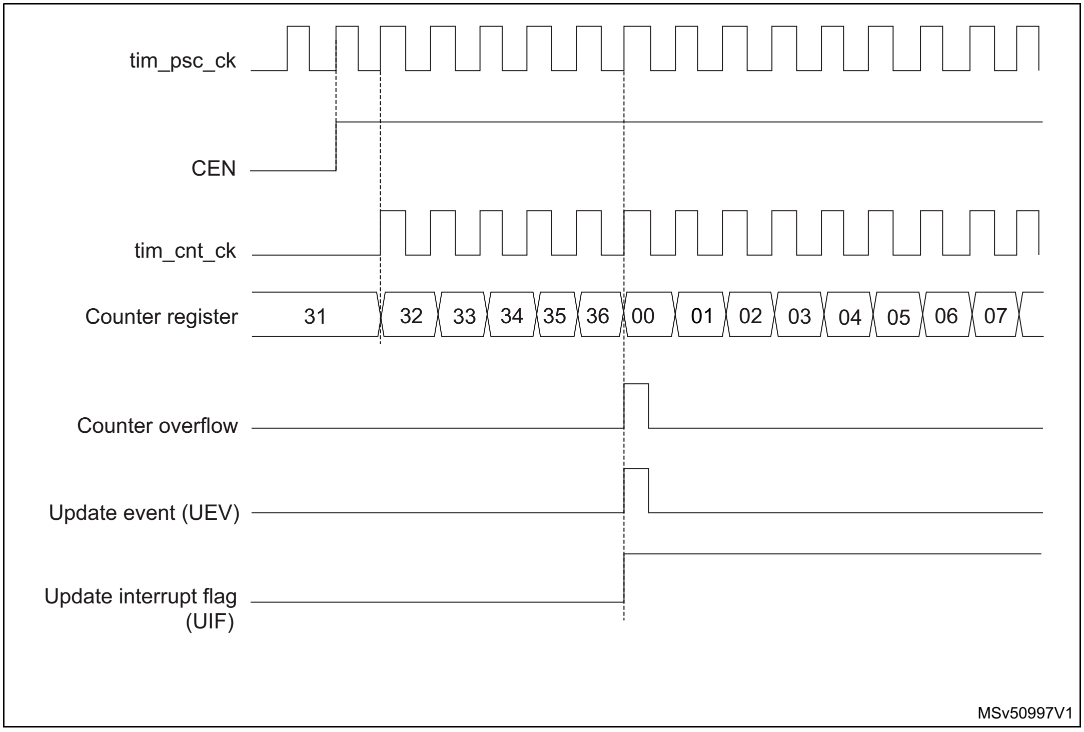


Figure 484. Counter timing diagram, internal clock divided by 2

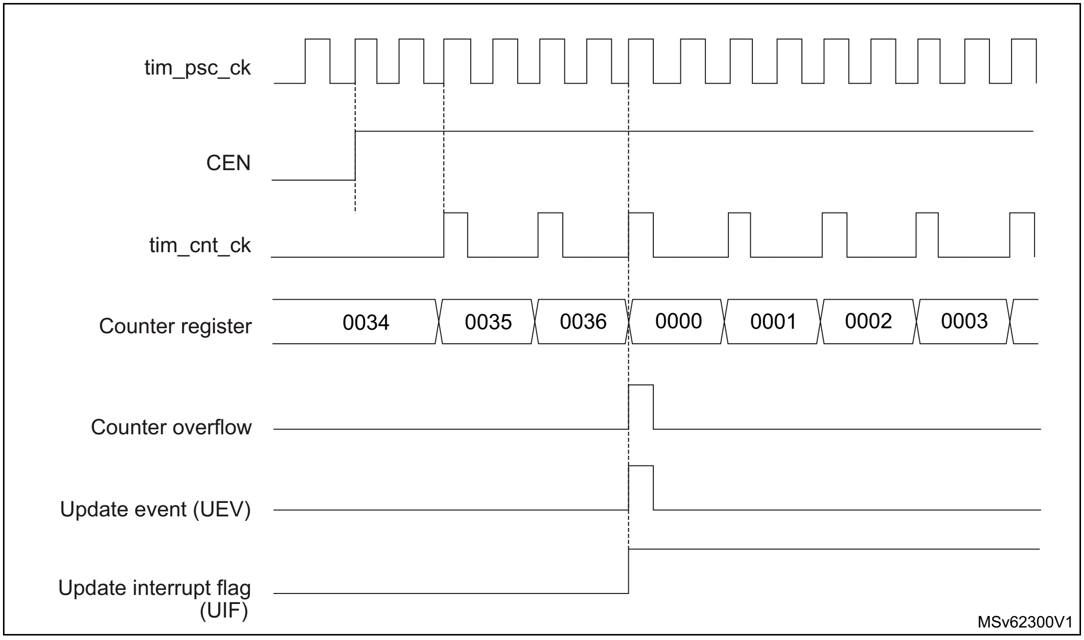


Figure 485. Counter timing diagram, internal clock divided by 4

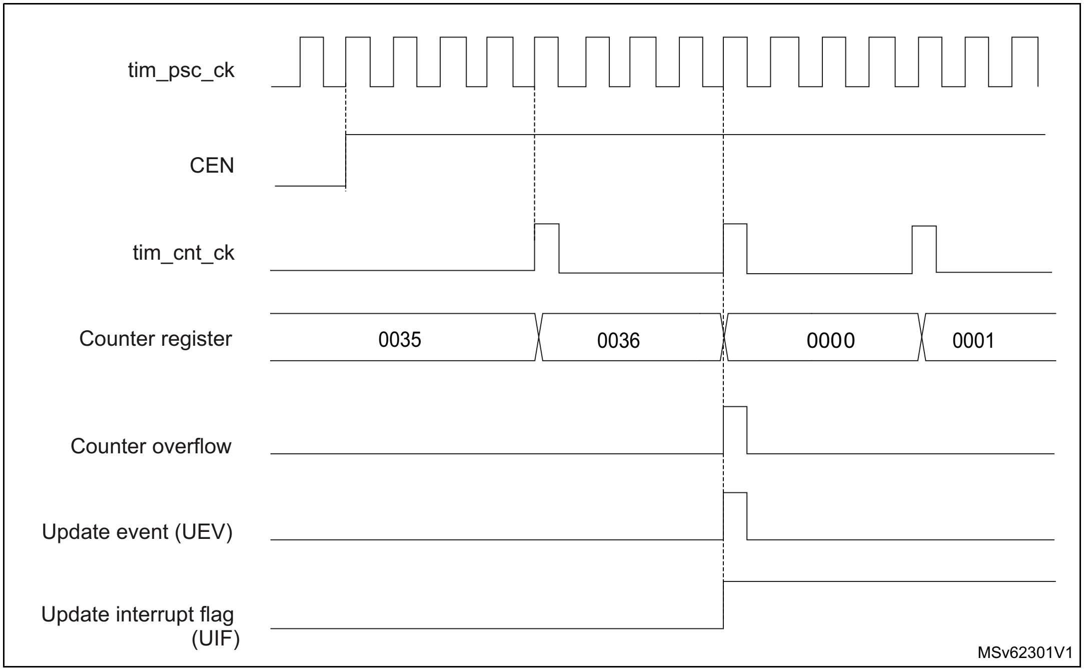


Figure 486. Counter timing diagram, internal clock divided by N

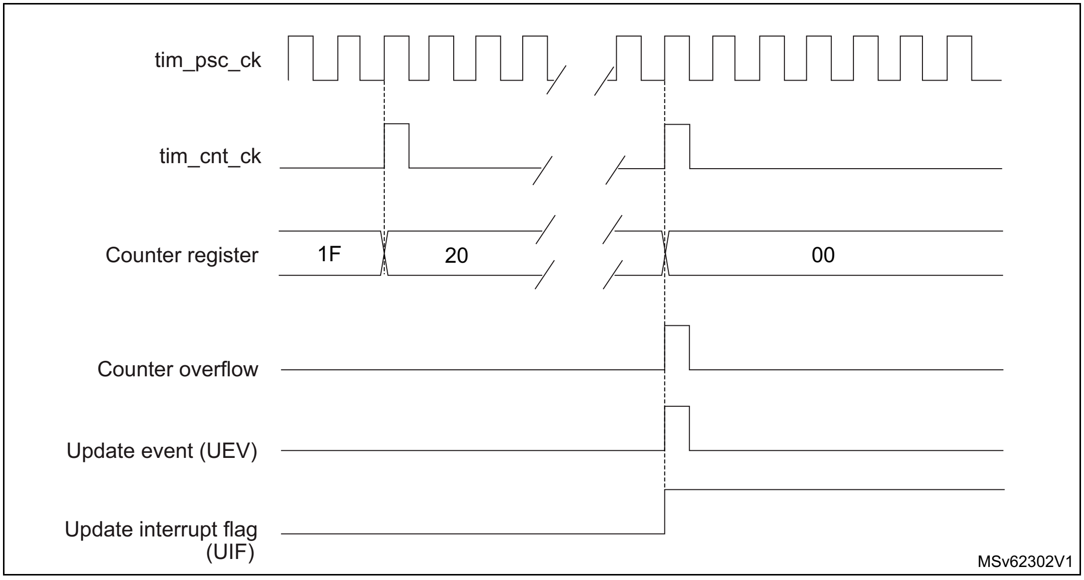


Figure 487. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

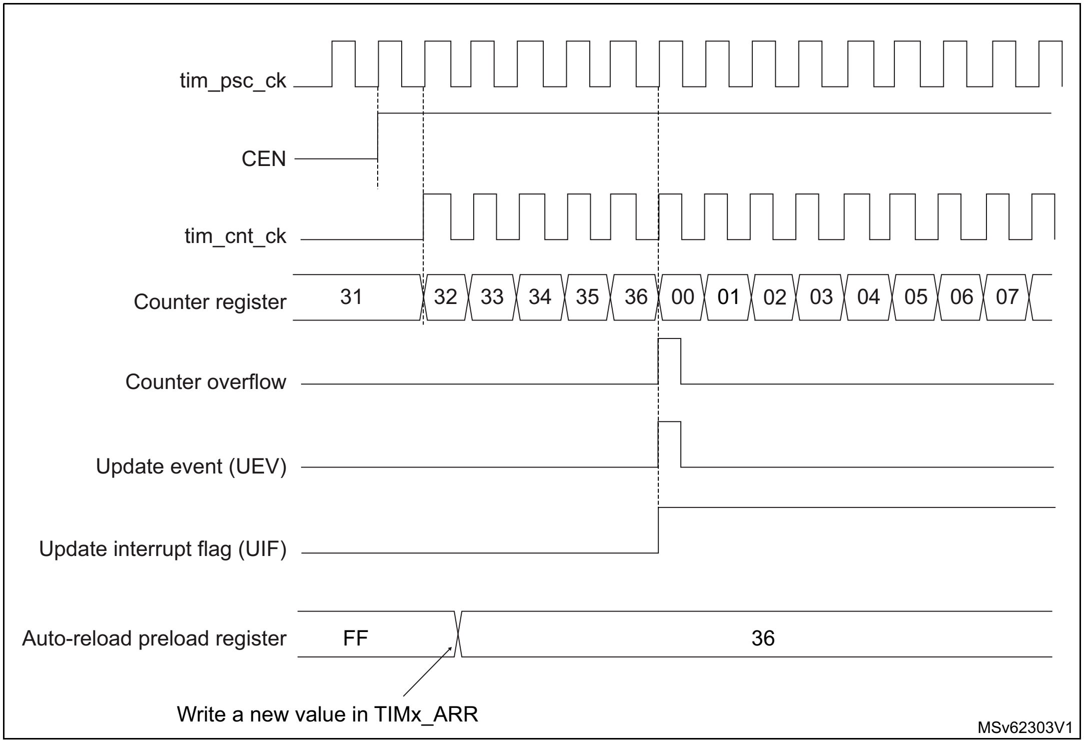
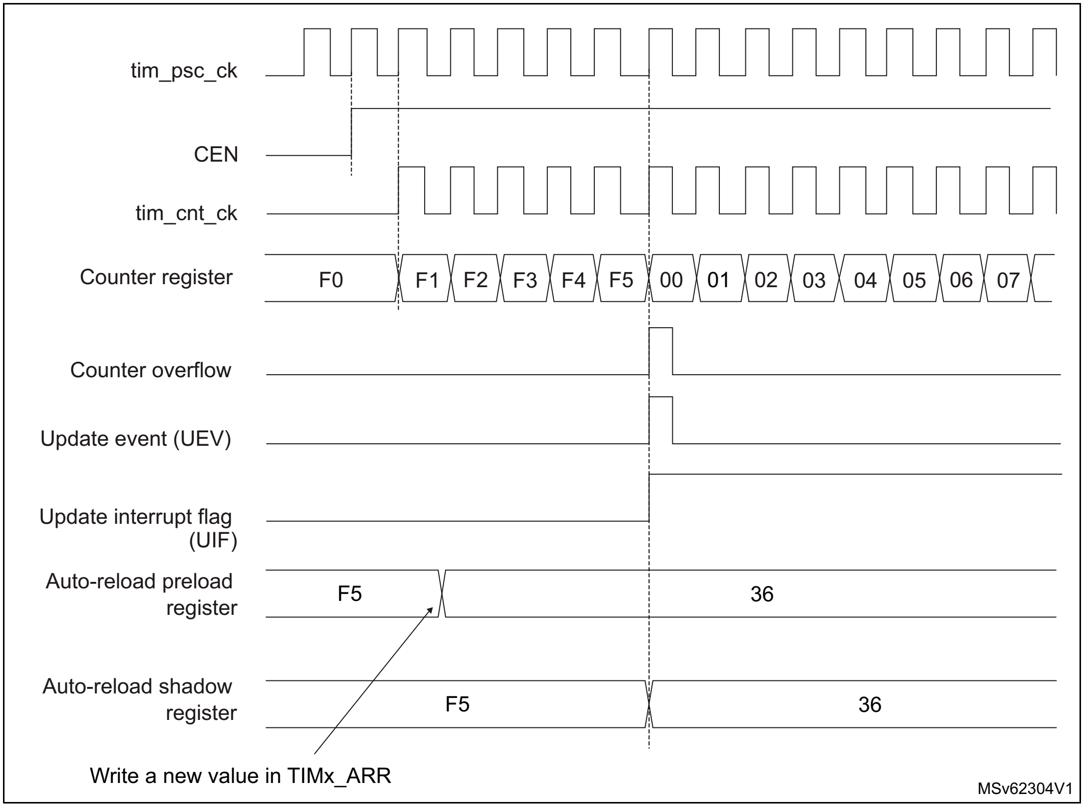


Figure 488. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



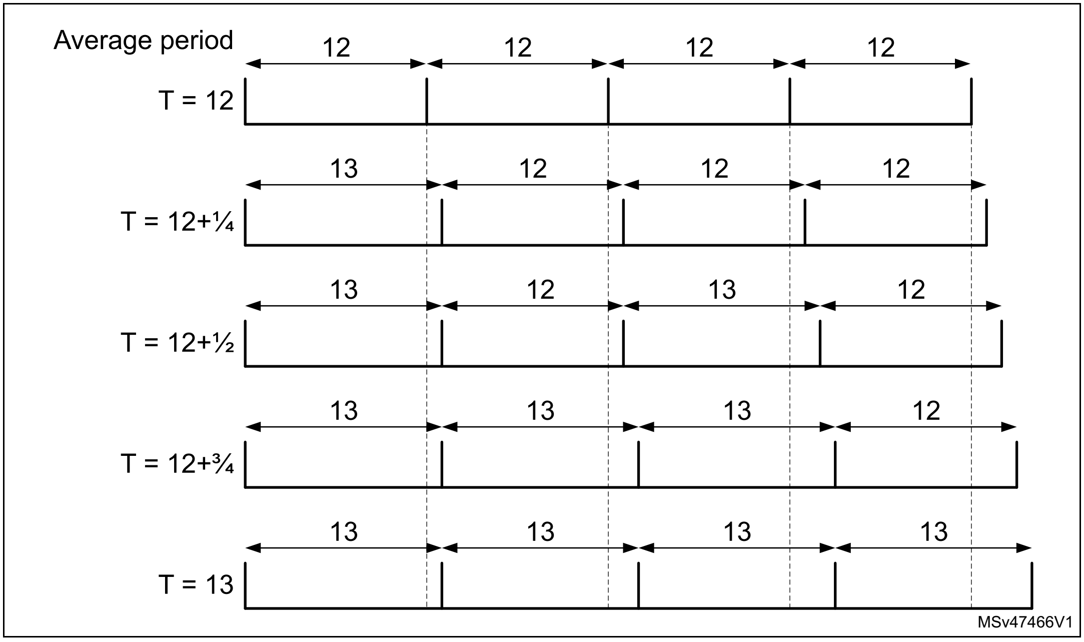
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

The *Figure 489* below presents the dithering principle applied to 4 consecutive counting periods.

Figure 489. Dithering principle

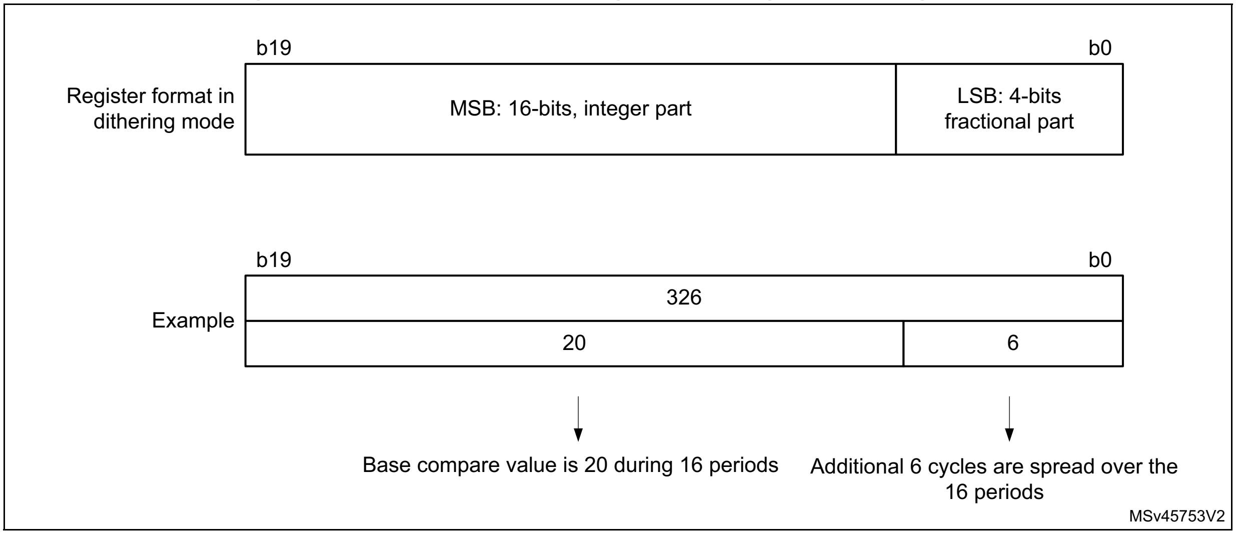


When the dithering mode is enabled, the register coding is changed as following (see *Figure 490* for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value

Note: The ARR values are updated automatically if the DITHEN bit is set / reset (for instance, if ARR= 0x05 with DITHEN=0, it is updated to ARR = 0x50 with DITHEN = 1).The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The DITHEN bit must be reset
4. The CEN bit can be set (eventually with ARPE = 1)

Figure 490. Data format and register coding in dithering mode


Register format in dithering mode
b19
b0
MSB: 16-bits, integer part
LSB: 4-bits fractional part
Example
b19
b0
326
20
6
Base compare value is 20 during 16 periods
Additional 6 cycles are spread over the 16 periods
MSv45753V2

The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{Cnt}} \Rightarrow F_{CntMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

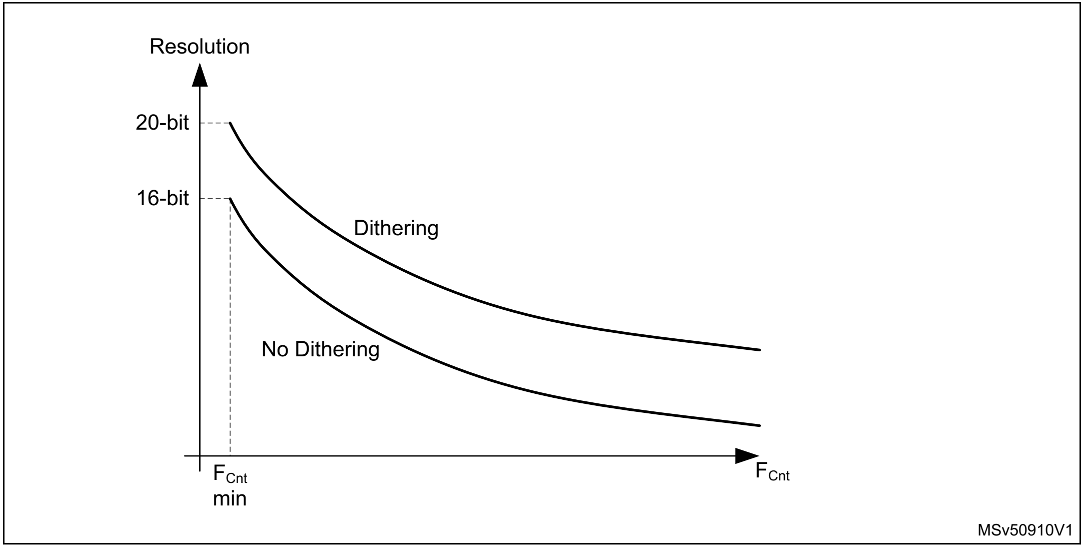
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

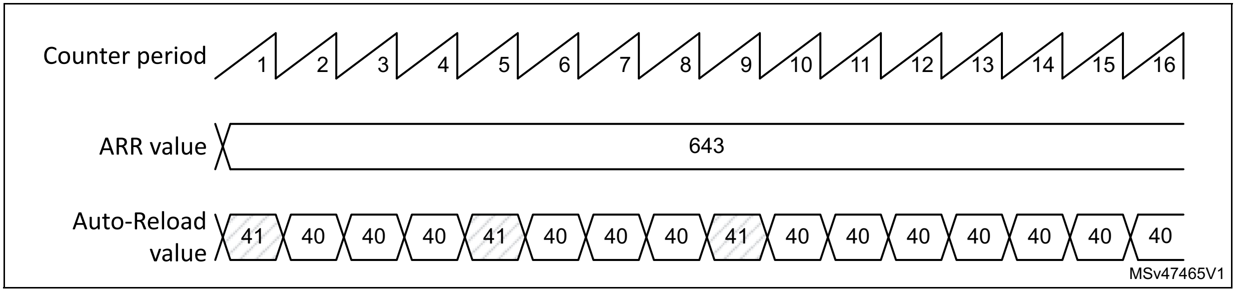
As shown on the *Figure 491* below, the dithering mode allows to increase the PWM resolution whatever the PWM frequency.

Figure 491. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in the *Figure 492* below.

Figure 492. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 304* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 304. TIMx_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

31.3.6 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 to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. 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.

31.3.7 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 305*.

Table 305. DMA request

DMA acronym	DMA request	Enable control bit
TIM6_UP TIM7_UP	Update	UDE

31.3.8 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

31.3.9 TIM6/TIM7 low-power modes

Table 306. Effect of low-power modes on TIM6/TIM7

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

31.3.10 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 307*.

Table 307. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7	Update	UIF	UIE	write 0 in UIF	Yes	No

31.4 TIM6/TIM7 registers

Refer to *Section 1.2 on page 73* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

31.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

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.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

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.

Bits 10:8 Reserved, must be kept at reset 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 CEN bit).

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 generates 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). 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

CEN is cleared automatically in one-pulse mode, when an update event occurs.

31.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

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.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

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

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used 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 a trigger output (tim_trgo).

001: **Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_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 when the CEN control bit is written.

010: **Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo 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.

Bits 3:0 Reserved, must be kept at reset value.

31.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

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.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.
1: Update DMA request enabled.

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

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.
1: Update interrupt enabled.

31.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

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:

– On counter overflow if UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

31.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

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.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

31.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

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

Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

31.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

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 $f_{tim\_cnt\_ck}$ is equal to $f_{tim\_psc\_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event.
(including when the counter is cleared through UG bit of TIMx_EGR register.

31.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	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.	ARR[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.
Refer to *Section 31.3.4: Time-base unit on page 1451* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

31.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 308. TIMx 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.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				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.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
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.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																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.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18- 0x20	Reserved																																
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.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.2 on page 81* for the register boundary addresses.

32 Low-power timer (LPTIM)

32.1 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.

32.2 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: PCLK or any of the embedded oscillators (see RCC section)
  - External clock source over LPTIM input (working with no embedded 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

32.3 LPTIM implementation

Table 309 describes LPTIM implementation on STM32G4 Series devices.

Table 309. STM32G4 Series LPTIM features

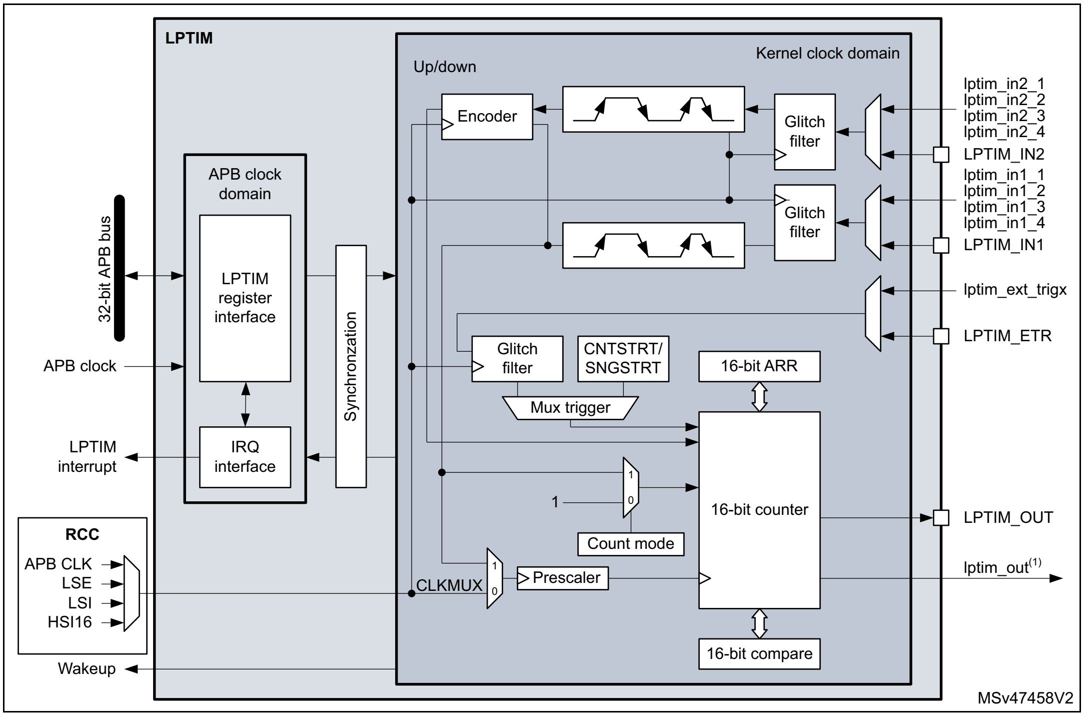
LPTIM modes/features[1]	LPTIM1
Encoder mode	X

1. X = supported.

32.4 LPTIM functional description

32.4.1 LPTIM block diagram

Figure 493. Low-power timer block diagram



1. lptim_out is the internal LPTIM output signal that can be connected to internal peripherals.

32.4.2 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter:

Table 310. LPTIM1 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO
lptim_ext_trig1	RTC_ALARMA
lptim_ext_trig2	RTC_ALARMB

Table 310. LPTIM1 external trigger connection (continued)

TRIGSEL	External trigger
lptim_ext_trig3	RTC_TAMP1_OUT
lptim_ext_trig4	RTC_TAMP2_OUT
lptim_ext_trig5	RTC_TAMP3_OUT
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT
lptim_ext_trig8	COMP3_OUT
lptim_ext_trig9	COMP4_OUT
lptim_ext_trig10	COMP5_OUT
lptim_ext_trig11	COMP6_OUT
lptim_ext_trig12	COMP7_OUT

Table 311. LPTIM1 input 1 connection

lptim_in1_mux	LPTIM1 input 1 connected to
lptim_in1_0	GPIO pin as LPTIM1_IN1 alternate function
lptim_in1_1	COMP1
lptim_in1_2	COMP3
lptim_in1_3	COMP5
lptim_in1_4	COMP7

Table 312. LPTIM1 input 2 connection

lptim_in2_mux	LPTIM1 input 2 connected to
lptim_in2_0	GPIO pin as LPTIM1_IN2 alternate function
lptim_in2_1	COMP2
lptim_in2_2	COMP4
lptim_in2_3	COMP6
lptim_in2_4	COMP6

32.4.3 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 PCLK (APB clock) or any other embedded oscillator 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 either from PCLK or any other embedded oscillator (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 will use 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.

32.4.4 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.

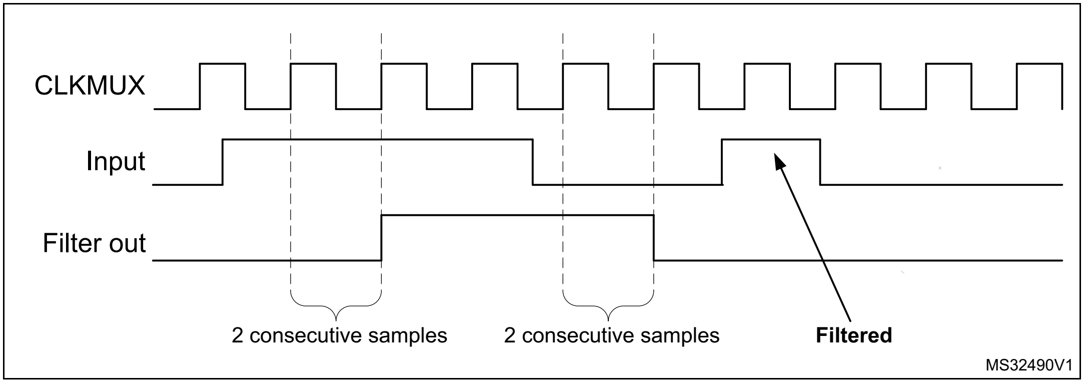
The digital filters are divided into two groups:

- The first group of digital filters protects the LPTIM external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal 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 494* shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

Figure 494. 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.

32.4.5 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 313. Prescaler division ratios

programming	dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

32.4.6 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 13 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 13 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 will be 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 will be 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.

32.4.7 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 reaching the ARR value.

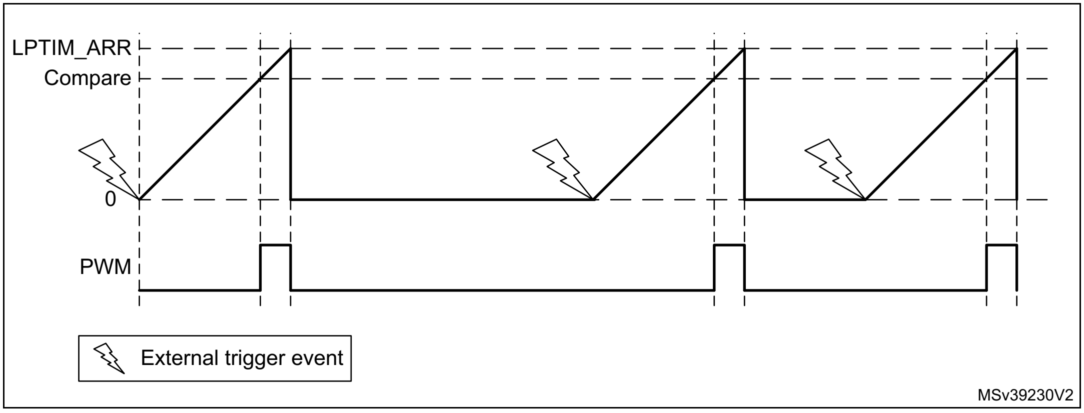
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event will re-start the timer. Any trigger event occurring after the counter starts and before the counter reaches ARR will be discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the counter register has stopped (contains zero value), will start the counter for a new one-shot counting cycle as shown in Figure 495.

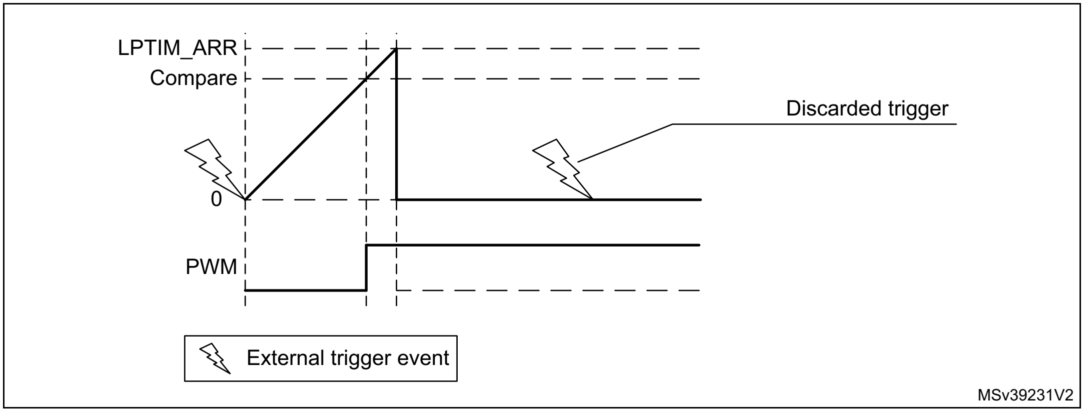
Figure 495. LPTIM output waveform, single counting mode configuration



- Set-once mode activated:

It should be noted that when the WAVE bit-field 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 496.

Figure 496. 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 will start the counter for one-shot counting.

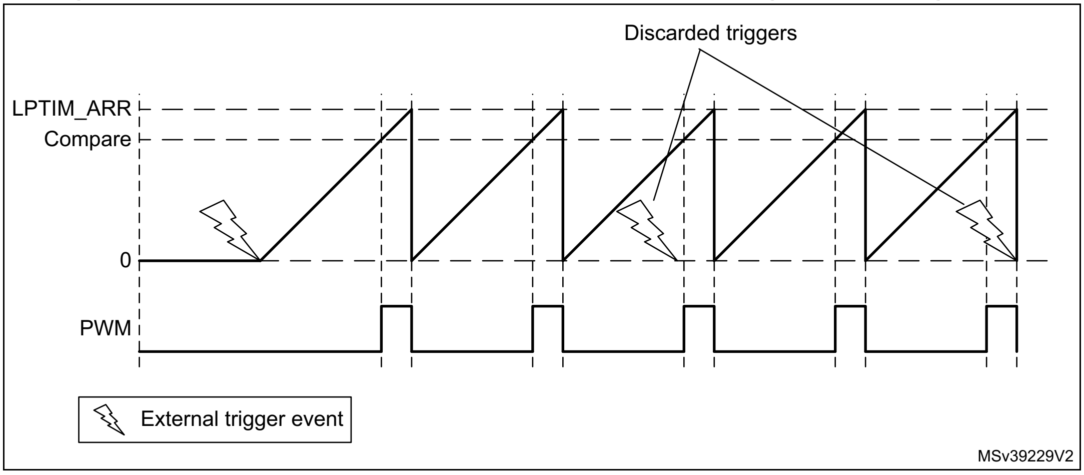
Continous mode

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 will start the counter for continuous counting. Any subsequent external trigger event will be discarded as shown in *Figure 497*.

In case of software start (TRIGEN[1:0] = ‘00’), setting CNTSTRT will start the counter for continuous counting.

Figure 497. 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 will switch the LPTIM to the One-shot mode. The counter (if active) will stop as soon as it reaches ARR.

If the One-shot mode was previously selected, setting CNTSTRT will switch the LPTIM to the Continuous mode. The counter (if active) will restart as soon as it reaches ARR.

32.4.8 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 will start the timer, any successive trigger event will reset the counter and the timer will restart.

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.

32.4.9 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.

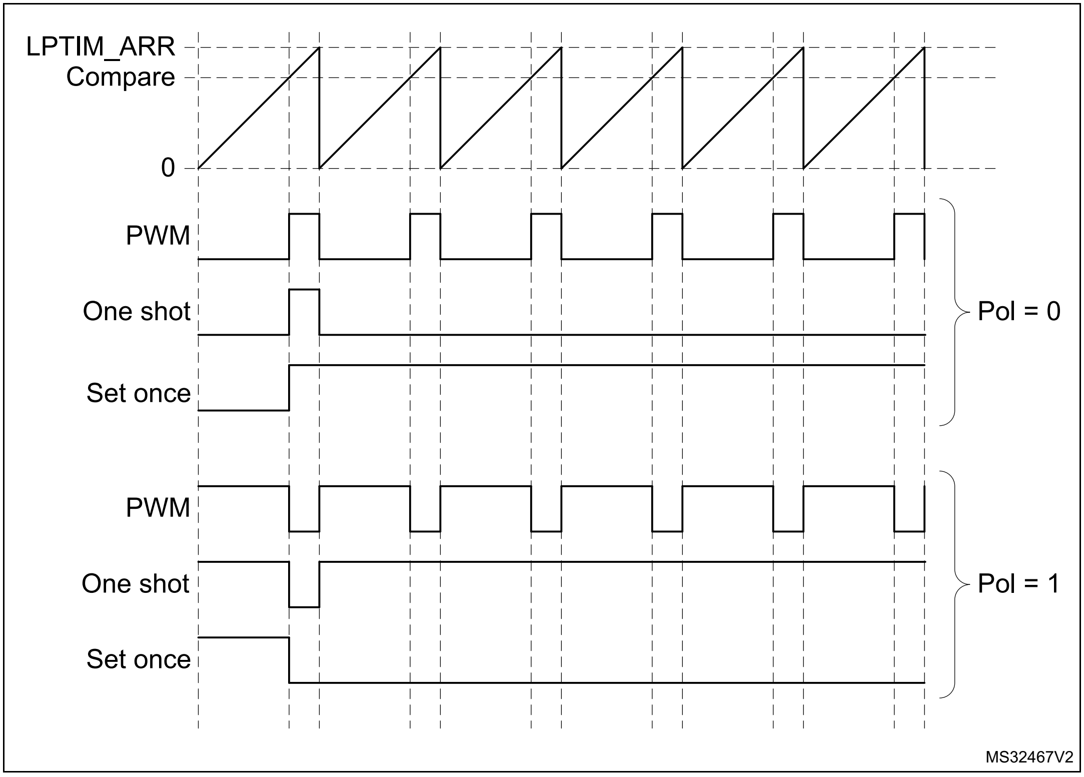
The LPTIM output waveform can be configured through the WAVE bit as follow:

- 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 will change 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 498* 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 498. Waveform generation



32.4.10 Register update

The LPTIM_ARR register and LPTIM_CMP register are updated immediately after the APB bus write operation, or at the end of the current period 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 the end of the current period, 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, will lead to unpredictable results.

32.4.11 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 will be 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.

32.4.12 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.

32.4.13 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 bit-field 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 will elapse before the reset is taken into account. This will make 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 will reset 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 will occur 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.

32.4.14 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. 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.

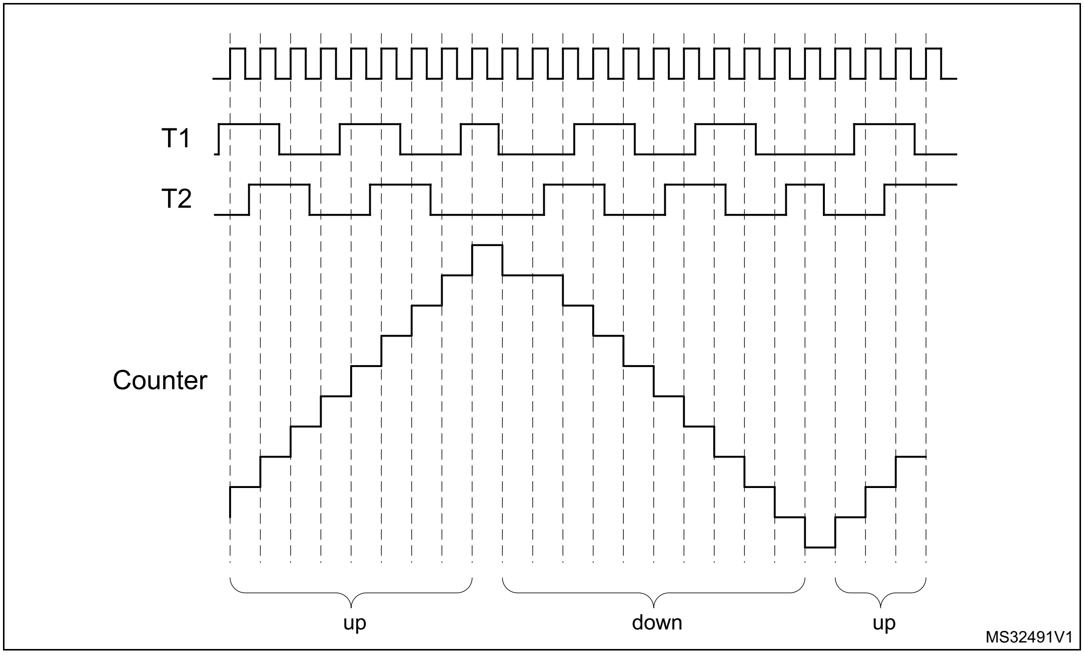
Table 314. Encoder counting scenarios

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 499. Encoder mode counting sequence



32.4.15 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.

32.5 LPTIM low-power modes

Table 315. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Low-power run	No effect.
Low-power sleep	No effect. LPTIM interrupts cause the device to exit the Low-power sleep mode.
Stop 0 / Stop 1	No effect when LPTIM is clocked by LSE or LSI. LPTIM interrupts cause the device to exit Stop 0 and Stop 1.
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

32.6 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).

Note: If any bit in the LPTIM_IER register (Interrupt Enable Register) is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.

Table 316. 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.

32.7 LPTIM registers

The peripheral registers can only be accessed by words (32-bit).

32.7.1 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.	Res.	Res.	DOWN	UP	ARR OK	CMP OK	EXT TRIG	ARRM	CMPM
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

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. Please refer to Section 32.3: 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. Please refer to Section 32.3: 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.

32.7.2 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.	Res.	Res.	DOWN CF	UPCF	ARRO KCF	CMPO KCF	EXTTR IGCF	ARRM CF	CMPM CF
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

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. Please refer to Section 32.3: LPTIM implementation.

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. Please refer to Section 32.3: LPTIM implementation.

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 CMPM flag in the LPTIM_ISR register

32.7.3 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.	Res.	Res.	DOWNI E	UPIE	ARRO KIE	CMPO KIE	EXT TRIGIE	ARRM IE	CMPM IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

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. Please refer to Section 32.3: LPTIM implementation.

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. Please refer to Section 32.3: LPTIM implementation.

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’)

32.7.4 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.	TRIG SEL [3]	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
		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**
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.

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. Please refer to Section 32.3: LPTIM implementation.

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 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 WAVEPOL bit controls the output polarity

0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CMP registers

1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CMP registers

Bit 20 **WAVE**: Waveform shape

The WAVE bit controls the output shape

0: Deactivate Set-once mode, PWM or One Pulse waveform depending on how the timer was started, CNTSTRT for PWM or SNGSTRT for One Pulse waveform.

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 will be ignored

1: A trigger event arriving when the timer is already started will reset and restart the 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 29, 15,14,13 **TRIGSEL[3:0]**: Trigger selector

The TRIGSEL bits select the trigger source that will serve as a trigger event for the LPTIM among the below 13 available sources:
0000: lptim_ext_trig0
0001: lptim_ext_trig1
0010: lptim_ext_trig2
0011: lptim_ext_trig3
0100: lptim_ext_trig4
0101: lptim_ext_trig5
0110: lptim_ext_trig6
0111: lptim_ext_trig7
1000: lptim_ext_trig8
1001: lptim_ext_trig9
1010: lptim_ext_trig10
1011: lptim_ext_trig11
1100: lptim_ext_trig12
Others: Reserved
See *Section 32.4.2: 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

If LPTIM is clocked by an external clock source:

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 32.4.14: Encoder mode* for more details about Encoder mode sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM will use:

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’).

32.7.5 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.	RST ARE	COUN TRST	CNT STRT	SNG STRT	ENA BLE
											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 will asynchronously reset LPTIM_CNT register content.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit will trigger 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).

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 will 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 will be 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 will stop at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It will be 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

32.7.6 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’).

32.7.7 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’).

32.7.8 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.
It should be noted that for a reliable LPTIM_CNT register read access, two consecutive read accesses must be performed and compared. A read access can be considered reliable when the values of the two consecutive read accesses are equal.

32.7.9 LPTIM option register (LPTIM_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.	IN2[2:1]		IN1[2:1]		IN2[0]	IN1[0]
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2[2:1]**: LPTimer input 2 remap extension

Set and cleared by software.
00: connected to COMP2
01: connected to COMP4
10: connected to COMP6
11: connected to COMP6

Bits 3:2 **IN1[2:1]**: LPTimer input 1 remap extension

Set and cleared by software.
00: connected to COMP1
01: connected to COMP3
10: connected to COMP5
11: connected to COMP7

Bit 1 **IN2[0]**: LPTimer input 2 remap

Set and cleared by software.
1: connected to COMP output according to IN2[2:1] value
0: connected to GPIO

Bit 0 **IN1[0]**: LPTimer input 1 remap

Set and cleared by software.
1: connected to COMP output according to IN1[2:1] value
0: connected to GPIO

32.7.10 LPTIM register map

The following table summarizes the LPTIM registers.

Table 317. 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.	Res.	Res.	DOWN[1]	UP[1]	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
	Reset value																										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.	Res.	Res.	DOWNCF[1]	UPCF[1]	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
	Reset value																										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.	Res.	Res.	DOWNIE[1]	UPIE[1]	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
	Reset value																										0	0	0	0	0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	TRIGSEL[3]	Res.	Res.	Res.	Res.	ENC[1]	COUNTMODE	PRELOAD	WAVEPOL	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	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	LPTIM_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.	IN2[2:1]		IN1[2:1]		IN2[0]	IN1[0]
	Reset value																											0	0	0	0	0	0

[1] *If LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 32.3: LPTIM implementation.*

Refer to *Section 2.2 on page 81* for the register boundary addresses.

33 Infrared interface (IRTIM)

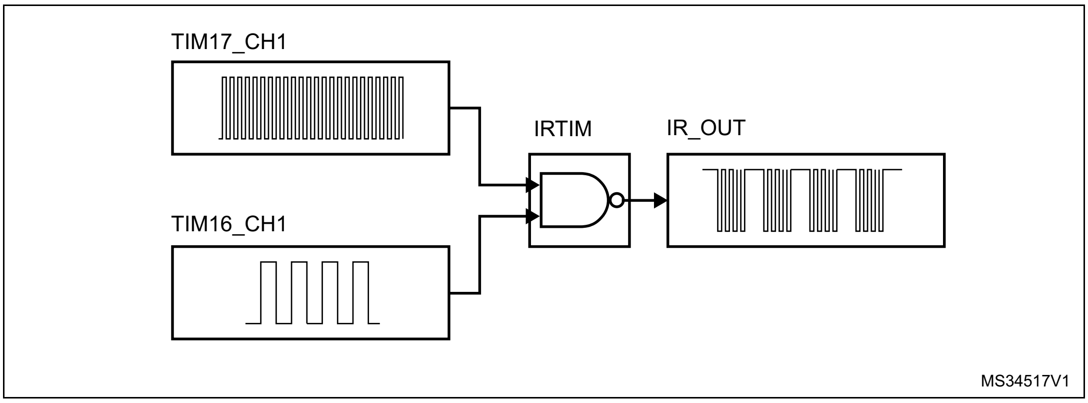
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16 and TIM17 as shown in *Figure 500*.

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 500. 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 and PA13 pins) 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.

34 AES hardware accelerator (AES)

34.1 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).

34.2 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 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

34.3 AES implementation

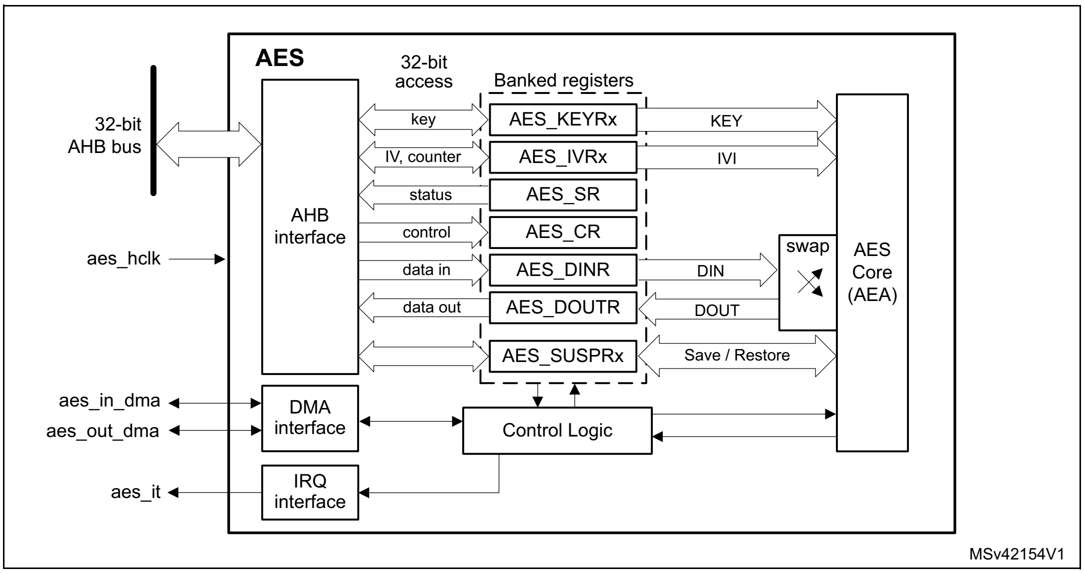
The devices have one AES peripheral.

34.4 AES functional description

34.4.1 AES block diagram

Figure 501 shows the block diagram of AES.

Figure 501. AES block diagram



34.4.2 AES internal signals

Table 318 describes the user relevant internal signals interfacing the AES peripheral.

Table 318. 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

34.4.3 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.
- **Mode 4:**
  ECB or CBC ciphertext single decryption using the key stored in the AES_KEYRx registers (the initial key is derived automatically).

Note: Mode 2 and mode 4 are only used when performing ECB and CBC decryption.

When Mode 4 is selected only one decryption can be done, therefore usage of Mode 2 and Mode 3 is recommended instead.

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 34.4.4: AES procedure to perform a cipher operation on page 1499*.

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)

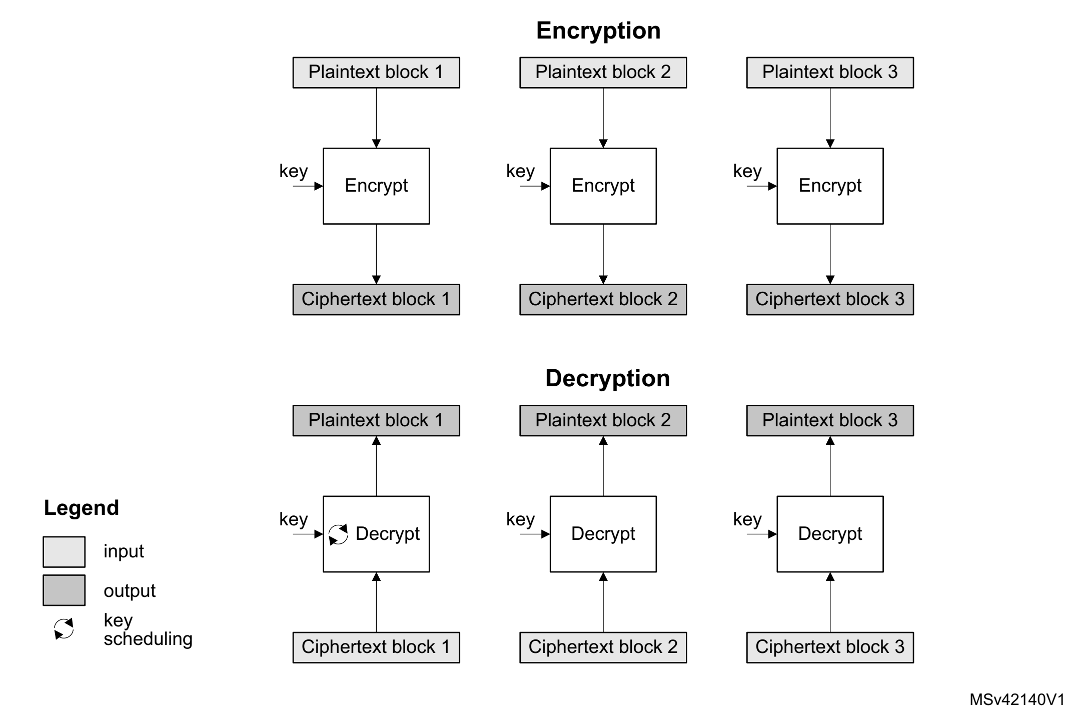
Note: *The chaining mode may be changed only when AES is disabled (bit EN of the AES_CR register cleared).*

Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from *Section 34.4.8: AES basic chaining modes (ECB, CBC)*.

Electronic codebook (ECB) mode

Figure 502. ECB encryption and decryption principle

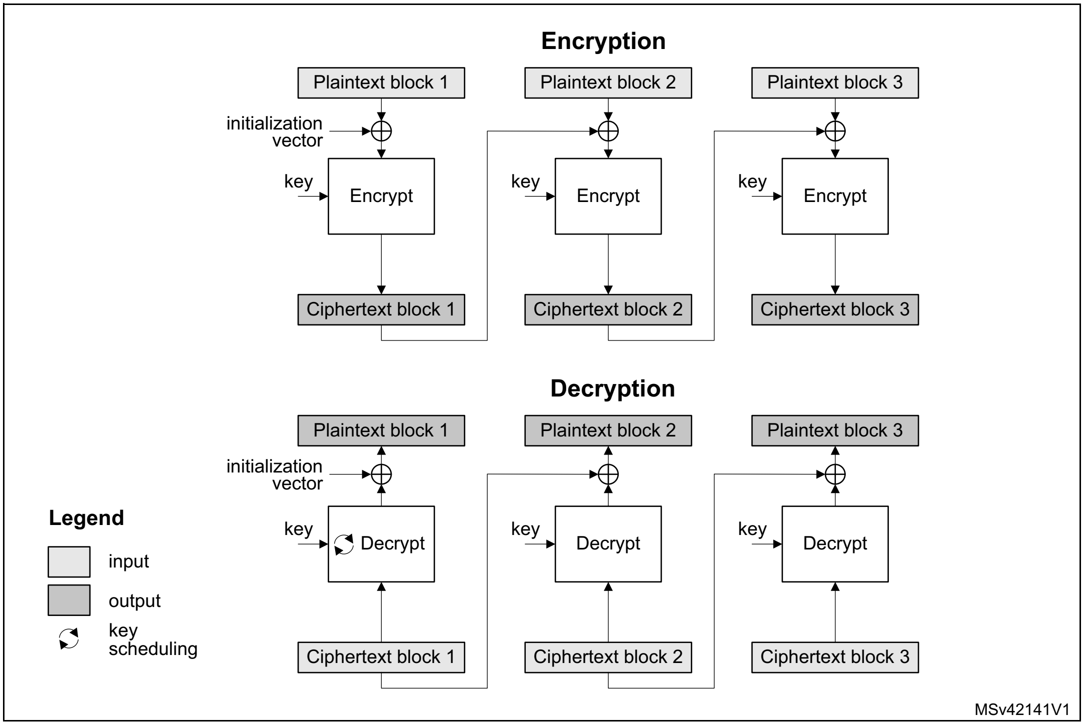


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: *For decryption, a special key scheduling is required before processing the first block.*

Cipher block chaining (CBC) mode

Figure 503. CBC encryption and decryption principle

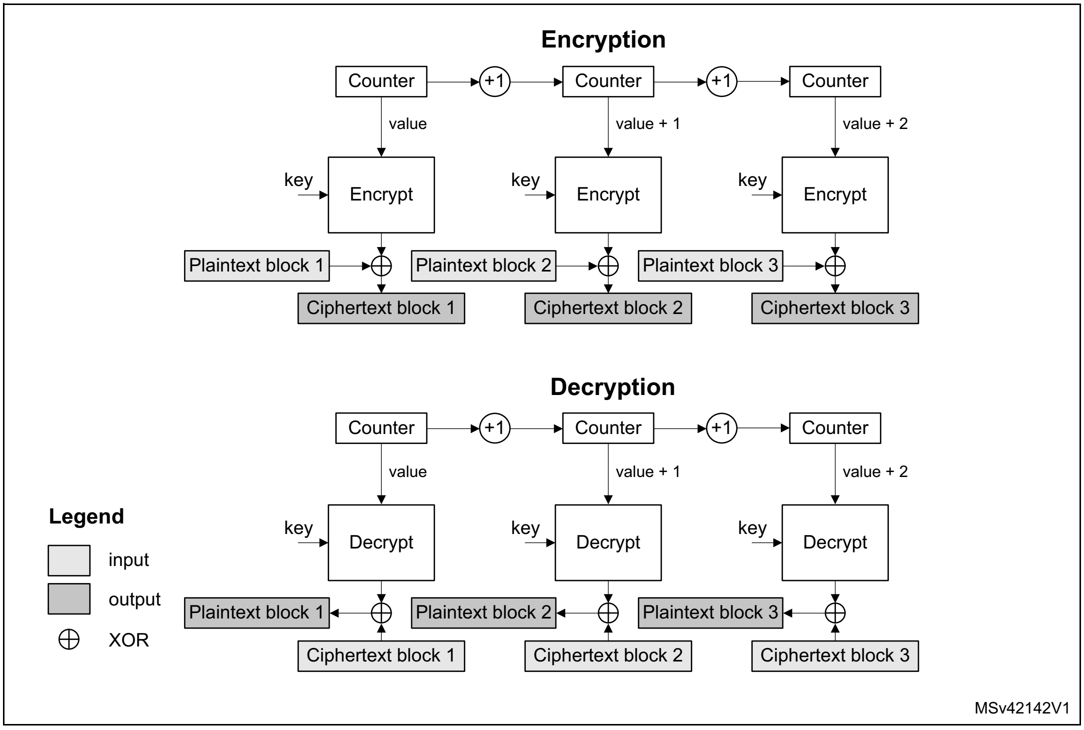


In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

Note: For decryption, a special key scheduling is required before processing the first block.

Counter (CTR) mode

Figure 504. CTR encryption and decryption principle

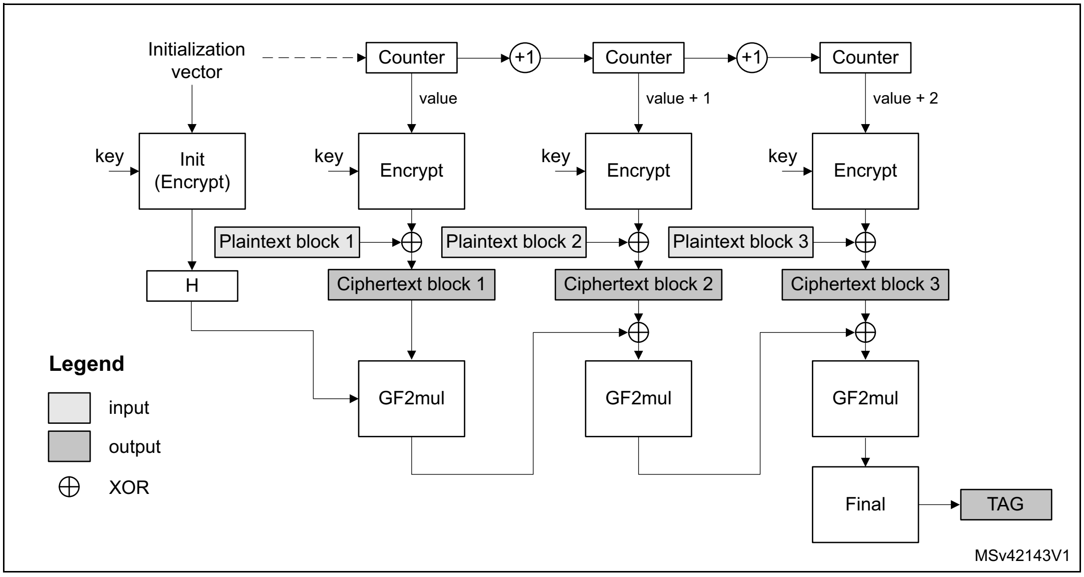


The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

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 505. GCM encryption and authentication principle

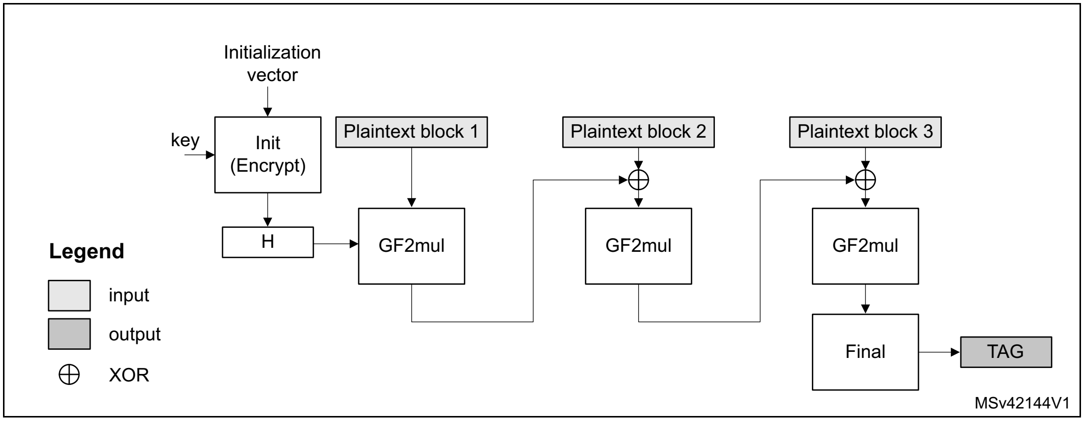


In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block 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 506. GMAC authentication principle

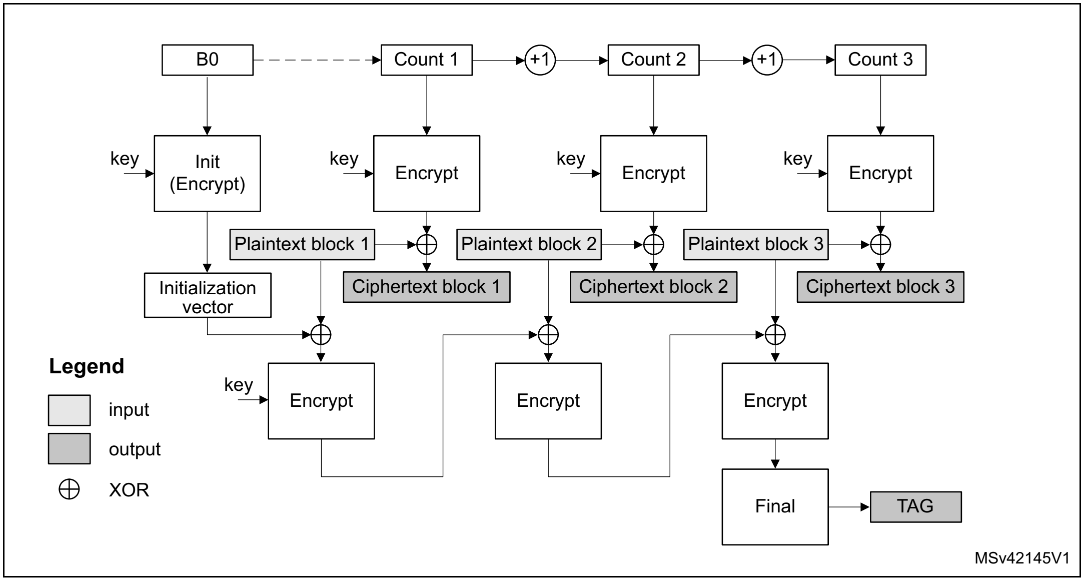


Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

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 507. 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.

34.4.4 AES procedure to perform a cipher operation

Introduction

A typical cipher operation is explained below. Detailed information is provided in sections starting from *Section 34.4.8: 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 34.4.5: 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 34.4.6: 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 34.4.16: 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.

34.4.5 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.

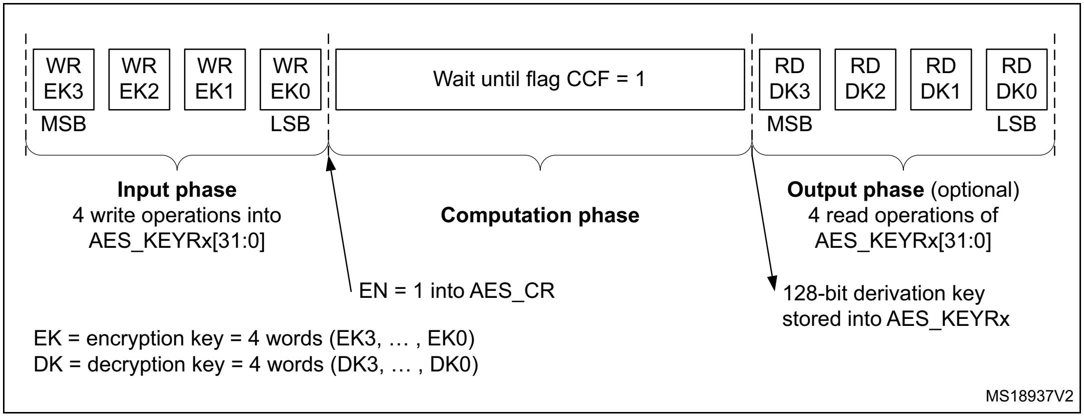
Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES_CR (key process only), then proceed with the decryption by setting MODE[1:0] to 10 (Mode 3, decryption only). Mode 2 usage is described below:

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, as shown in *Figure 508*. 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. Application can also read the AES_KEYRx register to obtain the derived key if needed, as shown in *Figure 508* (the processed key is loaded automatically into the AES_KEYRx registers).

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.

Figure 508. Encryption key derivation for ECB/CBC decryption (Mode 2)


WR EK3
WR EK2
WR EK1
WR EK0
MSB
LSB
Wait until flag CCF = 1
RD DK3
RD DK2
RD DK1
RD DK0
MSB
LSB
Input phase
4 write operations into AES_KEYRx[31:0]
Computation phase
Output phase (optional)
4 read operations of AES_KEYRx[31:0]
EN = 1 into AES_CR
128-bit derivation key stored into AES_KEYRx
EK = encryption key = 4 words (EK3, … , EK0)
DK = decryption key = 4 words (DK3, … , DK0)
MS18937V2

If the software stores the initial key prepared for decryption, it is enough to do the key schedule operation only once for all the data to be decrypted with a given cipher key.

Note: The operation of the key preparation lasts 59 or 82 clock cycles, depending on the key size (128- or 256-bit).

34.4.6 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 34.4.4: AES procedure to perform a cipher operation on page 1499* 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 34.4.13: AES data registers and data swapping for details).

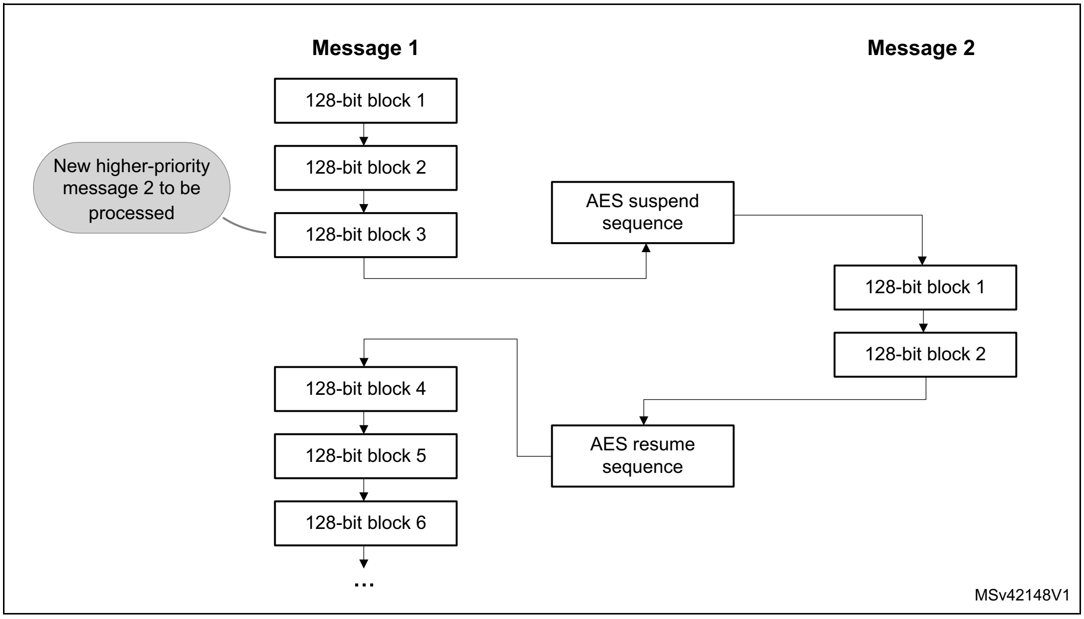
34.4.7 AES task suspend and resume

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 509 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 509. Example of suspend mode management



A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

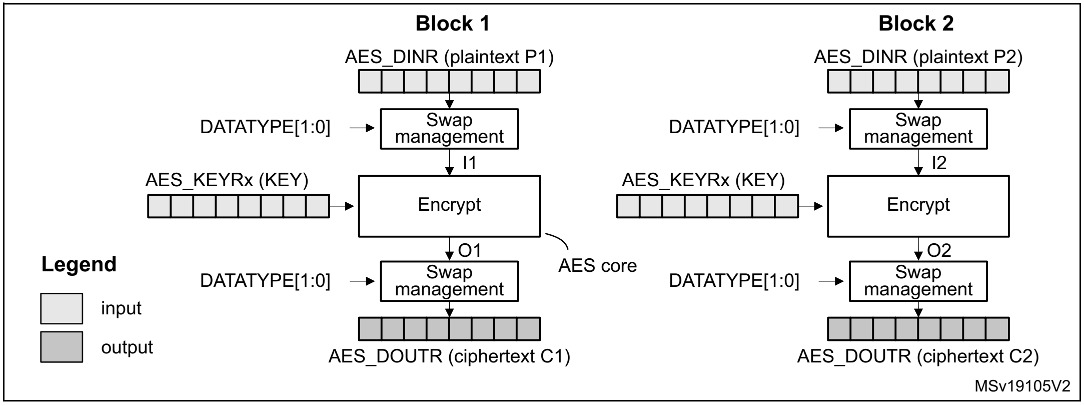
34.4.8 AES basic chaining modes (ECB, CBC)

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 510 illustrates the electronic codebook (ECB) encryption.

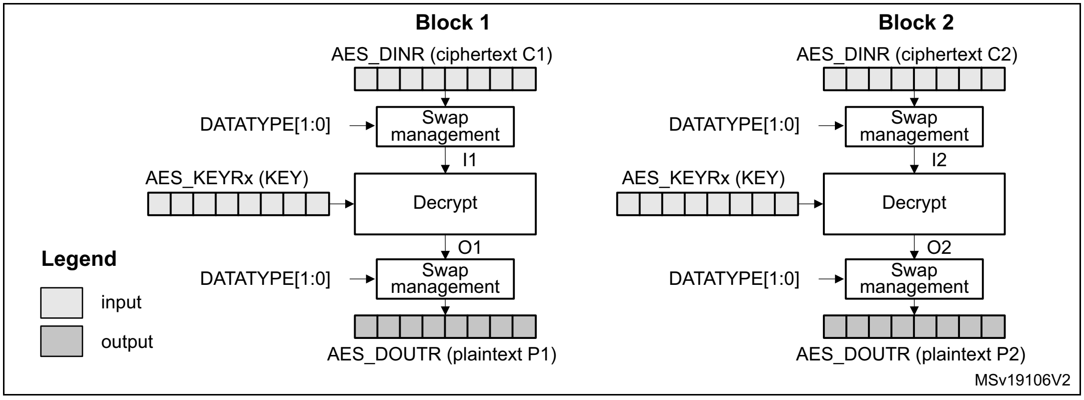
Figure 510. ECB encryption



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 511 illustrates the electronic codebook (ECB) decryption.

Figure 511. ECB decryption

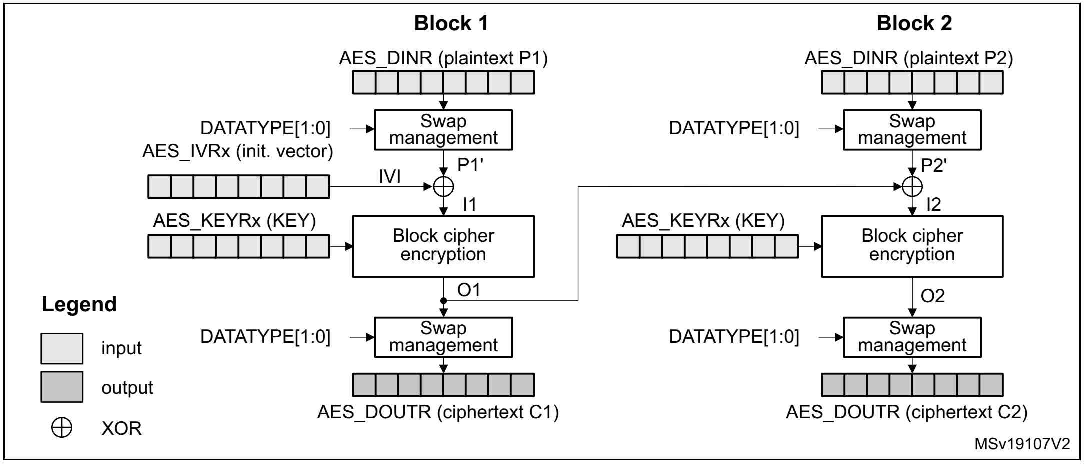


To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported 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 512 illustrates the cipher block chaining (CBC) encryption.

Figure 512. CBC encryption

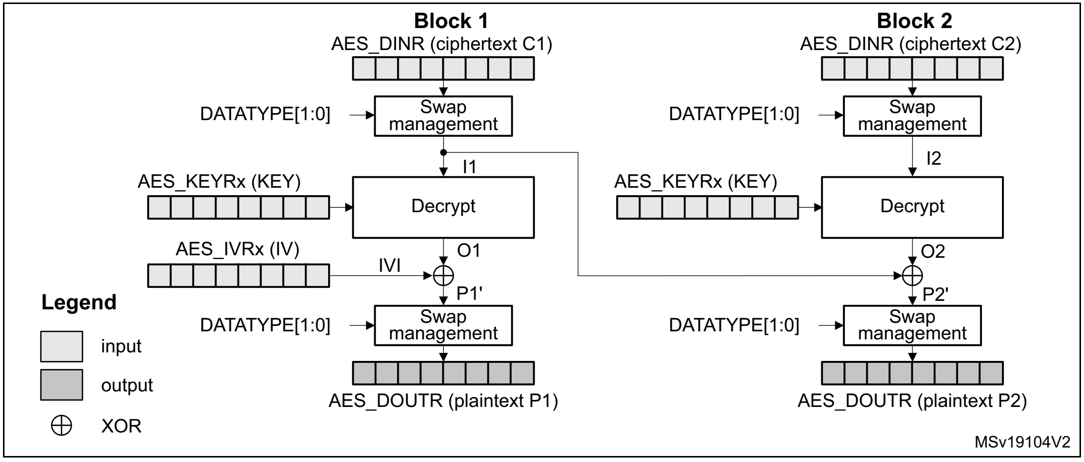


In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1’), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after 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 34.4.6: AES ciphertext stealing and data padding*.

Figure 513 illustrates the cipher block chaining (CBC) decryption.

Figure 513. 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 34.4.6: AES ciphertext stealing and data padding*.

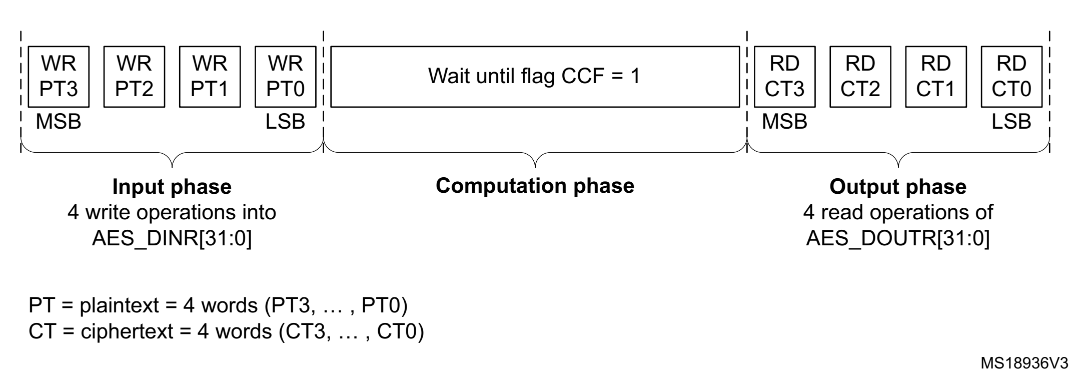
For more information on data swapping, refer to *Section 34.4.13: AES data registers and data swapping*.

ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in *Section 34.4.4*):

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 514*.
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 514*. 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 514. ECB/CBC encryption (Mode 1)

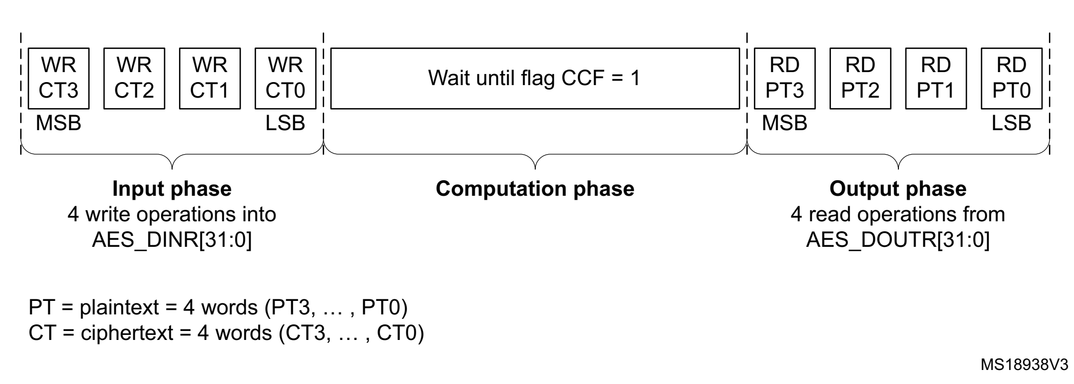

WR PT3
WR PT2
WR PT1
WR PT0
MSB
LSB
Wait until flag CCF = 1
RD CT3
RD CT2
RD CT1
RD CT0
MSB
LSB
Input phase
4 write operations into AES_DINR[31:0]
Computation phase
Output phase
4 read operations of AES_DOUTR[31:0]
PT = plaintext = 4 words (PT3, … , PT0)
CT = ciphertext = 4 words (CT3, … , CT0)
MS18936V3

ECB/CBC decryption sequence

The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in *Section 34.4.4*).

1. Follow the steps described in *Section 34.4.5: 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 515*.
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 515*. 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 515. ECB/CBC decryption (Mode 3)


WR CT3
WR CT2
WR CT1
WR CT0
MSB
LSB
Wait until flag CCF = 1
RD PT3
RD PT2
RD PT1
RD PT0
MSB
LSB
Input phase
4 write operations into AES_DINR[31:0]
Computation phase
Output phase
4 read operations from AES_DOUTR[31:0]
PT = plaintext = 4 words (PT3, … , PT0)
CT = ciphertext = 4 words (CT3, … , CT0)
MS18938V3

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).

Note: *In point 7, the derived key information stored in AES_KEYRx registers can optionally be saved in memory if the interrupted process is a decryption. Otherwise those registers do not need to be saved as the original key value is known by the application*

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. In case of decryption, derived key information can be written in AES_KEYRx register instead of the original key value.
4. Prepare the decryption key as described in *Section 34.4.5: AES decryption round key preparation* (only required for ECB or CBC decryption). This step is not necessary if derived key information is loaded in AES_KEYRx registers.
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.

Alternative single ECB/CBC decryption using Mode 4

The sequence of events to perform a single round of ECB/CBC decryption using Mode 4 is:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select the Mode 4 by setting to 11 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 0x0 or 0x1, respectively.
3. Select key length of 128 or 256 bits via KEYSIZE bitfield of the AES_CR register.
4. Write the AES_KEYRx registers with the encryption key. Write the AES_IVRx registers if the CBC mode is 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 cipher text (MSB first).
7. Wait until the CCF flag is set in the AES_SR register.
8. Read the AES_DOUTR register four times to get the plain text (MSB first). Then clear the CCF flag by setting the CCFC bit of the AES_CR register.

Note: *When mode 4 is selected mode 3 cannot be used.*

In mode 4, the AES_KEYRx registers contain the encryption key during all phases of the processing. No derivation key is stored in these registers. It is stored internally in AES.

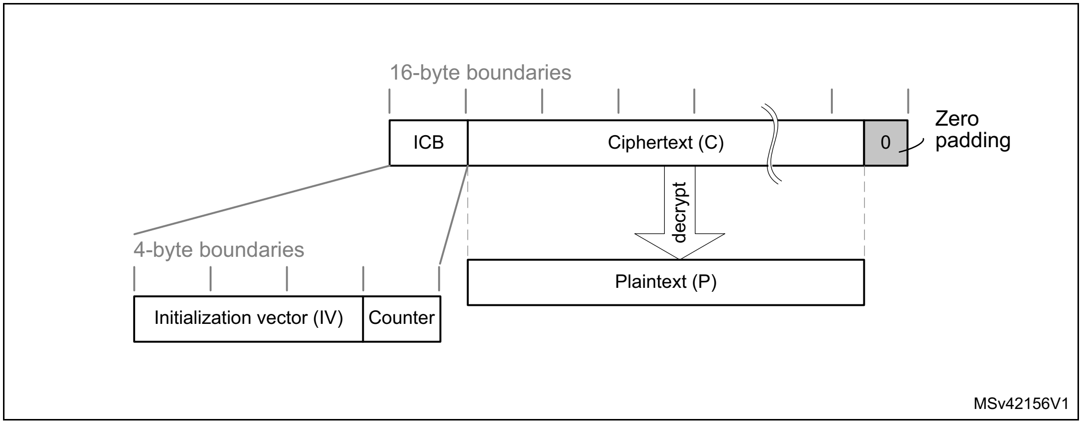
34.4.9 AES counter (CTR) mode

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 516*.

Figure 516. 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 517 and *Figure 518* 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 517. CTR encryption

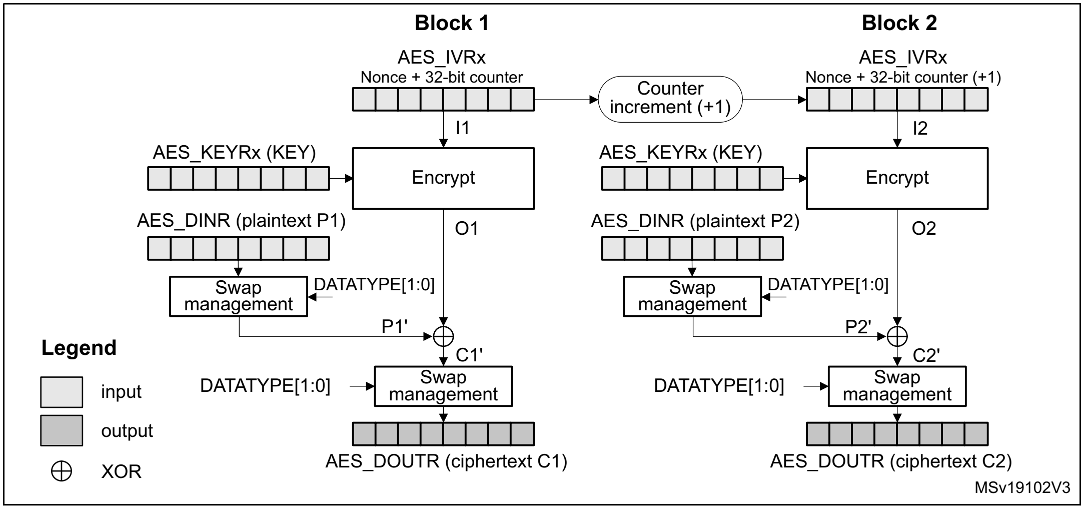
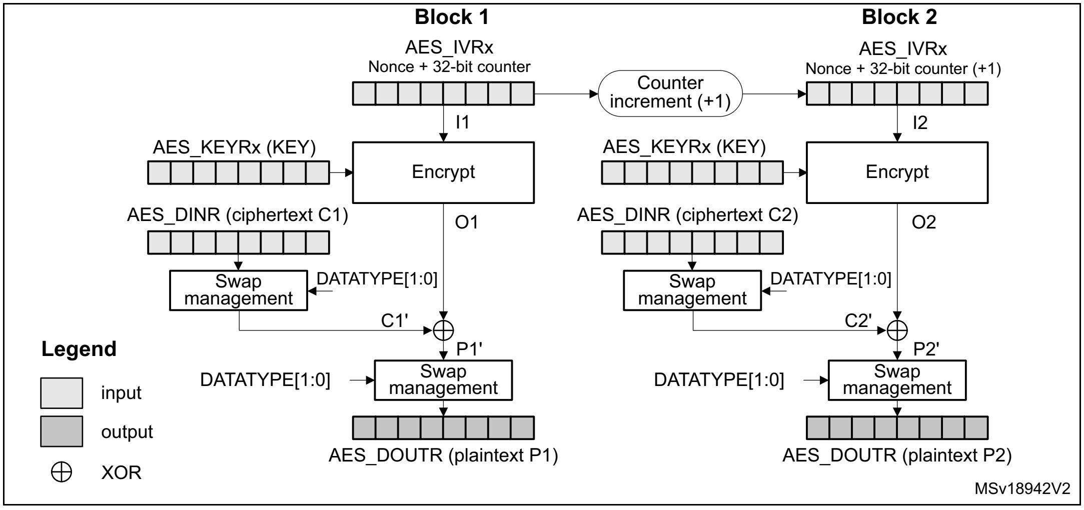


Figure 518. CTR decryption



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 319*.

Table 319. 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 319*.
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 34.4.4: 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 34.4.8: AES basic chaining modes (ECB, CBC)*.

Note: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.

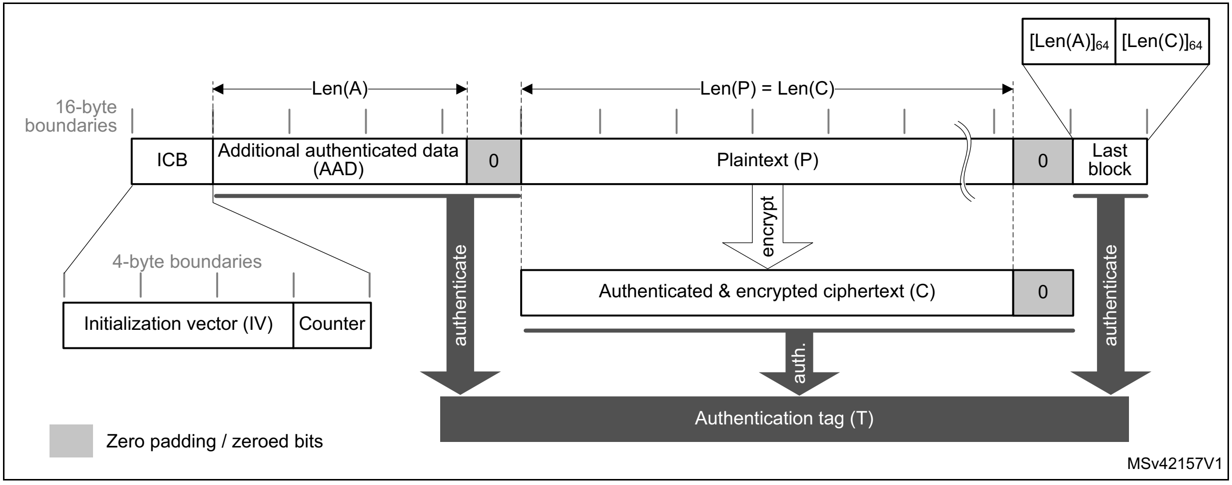
34.4.10 AES Galois/counter mode (GCM)

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 519*.

Figure 519. 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 320*.

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.

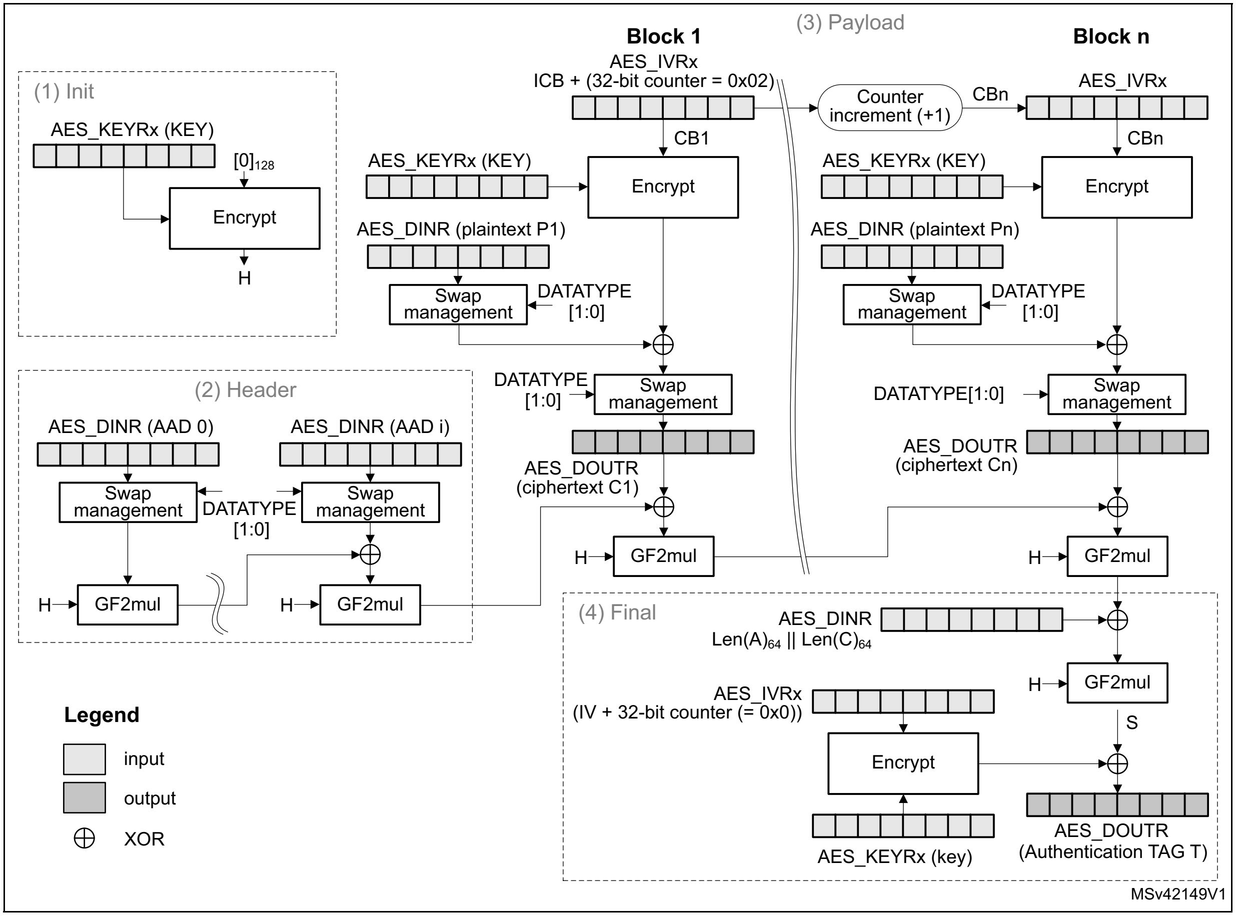
Table 320. GCM last block definition

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 520 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 520. 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 321*).

Table 321. Initialization of SAES_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 settings 01 and 11 of the MODE[1:0] bitfield are 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 321*.
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 34.4.4: 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 34.4.4: AES procedure to perform a cipher operation on page 1499*, 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 320*. 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.

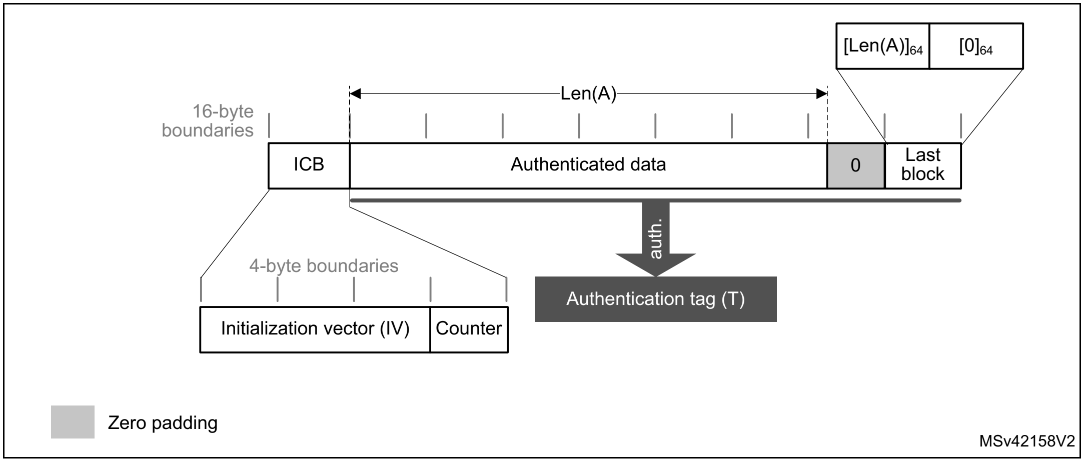
34.4.11 AES Galois message authentication code (GMAC)

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 521*.

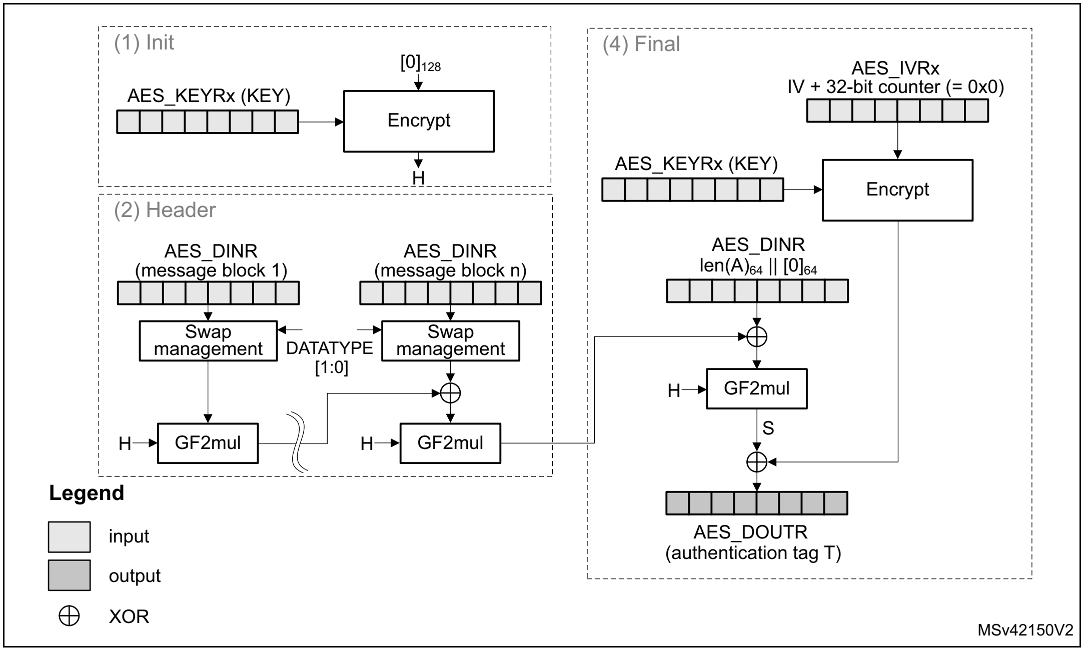
Figure 521. Message construction in GMAC mode



AES GMAC processing

Figure 522 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 522. 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.

34.4.12 AES counter with CBC-MAC (CCM)

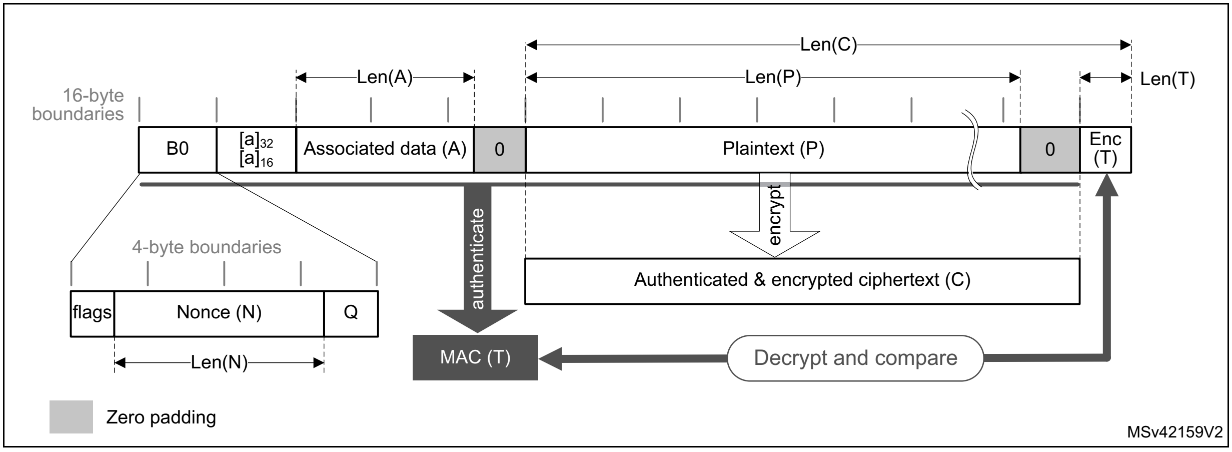
Overview

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 523*.

Figure 523. 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 523*). 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 523*).
- **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

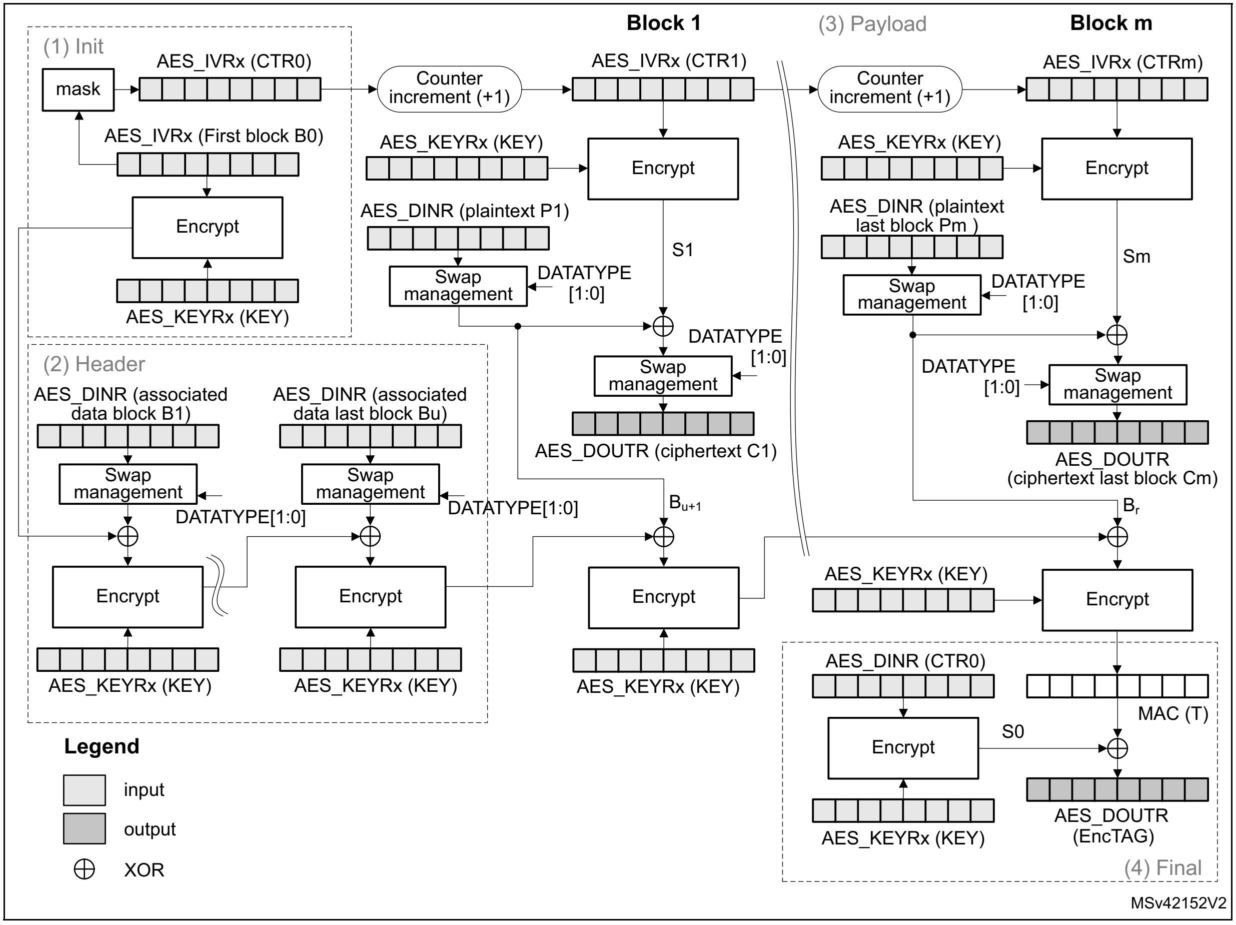
CTR1: 0710111213 141516 00000000 00000001

Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.

CCM processing

Figure 524 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 524. 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 322 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 322. 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 settings 01 and 11 of the MODE[1:0] bitfield are 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 322*.
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 34.4.4: 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 34.4.4: AES procedure to perform a cipher operation on page 1499*, 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.

34.4.13 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 34.4.16: AES DMA interface*.

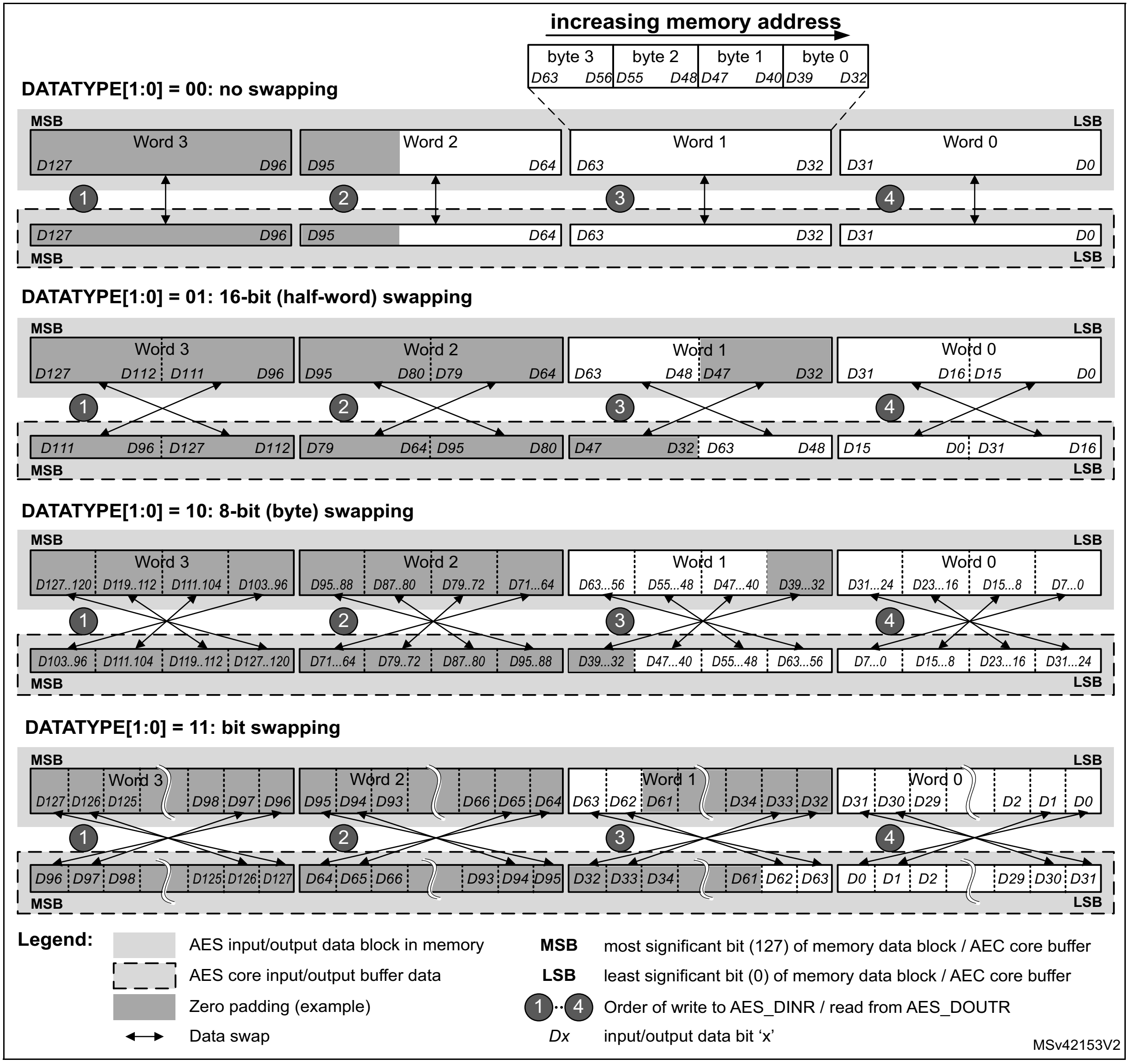
Data swapping

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 525* 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 525. 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 525 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

34.4.14 AES key registers

The AES_KEYRx registers store the encryption or decryption key bitfield KEY[127:0] or KEY[255:0]. 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 key is spread over eight registers as shown in *Table 323*.

Table 323. 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.

34.4.15 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.

34.4.16 AES DMA interface

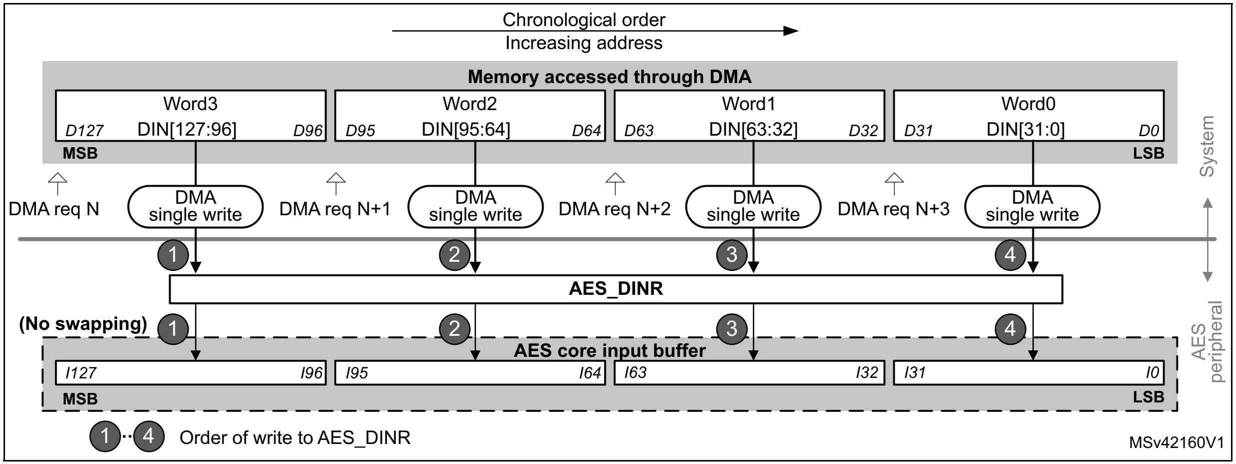
The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the AES_CR register.

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 526*.

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 34.4.4: AES procedure to perform a cipher operation.

Figure 526. DMA transfer of a 128-bit data block during input phase

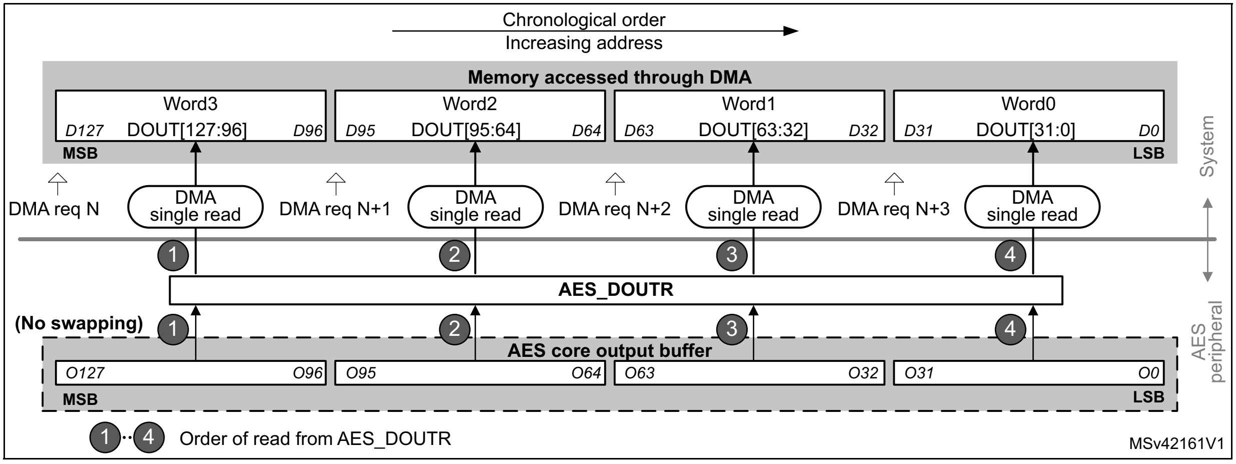


Data output using DMA

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 527*.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Figure 527. 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 is not relevant and can be ignored (left set) by software. It must only be cleared when transiting back to data transferring managed by software. See *Suspend/resume operations in ECB/CBC modes* in *Section 34.4.8: AES basic chaining modes (ECB, CBC)* as example.

34.4.17 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 34.5: 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 34.5: AES interrupts*.

The WRERR flag is cleared by setting the ERRC bit of the AES_CR register.

34.5 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 324 gives a summary of the interrupt sources, their event flags and enable bits.

Table 324. 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.

34.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

Table 325. 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
	Mode 4: Key derivation then decryption	ECB, CBC	106
256-bit	Mode 1: Encryption	ECB, CBC, CTR	75
	Mode 2: Key derivation	-	82
	Mode 3: Decryption	ECB, CBC, CTR	75
	Mode 4: Key derivation then decryption	ECB, CBC	145

Table 326. 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).

34.7 AES registers

34.7.1 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).
Use of DMA with Mode 4 (single decryption) is not recommended.

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).
Use of DMA with Mode 4 (single decryption) is not recommended.

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: Mode 4: key derivation then single decryption
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. Any attempt to selecting Mode 4 while either ECB or CBC chaining mode is not selected, defaults to effective selection of Mode 3. It is not possible to select a Mode 3 following a Mode 4.

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 34.4.13: 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.

34.7.2 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.

34.7.3 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) and **Mode 4** (key derivation then single decryption): ciphertext

The data swap operation is described in *Section 34.4.13: AES data registers and data swapping on page 1523*.

34.7.4 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 (AES_KEYRx registers used for output)
- **Mode 3** (decryption) and **Mode 4** (key derivation then single decryption): plaintext

The data swap operation is described in *Section 34.4.13: AES data registers and data swapping on page 1523*.

34.7.5 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]															
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
KEY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

This 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) and **Mode 4** (key derivation then single decryption): 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. After writing the encryption key into the bitfield, its reading before enabling AES returns the same value. Its reading after enabling AES and after the CCF flag is set returns the decryption key derived from the encryption key.

Note: In mode 4 (key derivation then single decryption) the bitfield always contains the encryption key.

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 34.4.14: AES key registers on page 1525* for more details.

34.7.6 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]															
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
KEY[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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.

34.7.7 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]															
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
KEY[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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.

34.7.8 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]															
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
KEY[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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.

34.7.9 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 34.4.15: AES initialization vector registers on page 1525* 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

34.7.10 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.

34.7.11 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.

34.7.12 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.

34.7.13 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]															
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
KEY[143:128]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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.

34.7.14 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]															
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
KEY[175:160]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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.

34.7.15 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]															
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**
KEY[207:192]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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.

34.7.16 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]															
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**
KEY[239:224]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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).

34.7.17 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.

34.7.18 AES register map

Table 327. 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

Table 327. 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
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
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 2.2 on page 81* for the register boundary addresses.

35 Real-time clock (RTC)

35.1 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.

35.2 RTC main features

The RTC supports the following features (see *Figure 528: RTC block diagram*):

- Calendar with subsecond, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- 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.

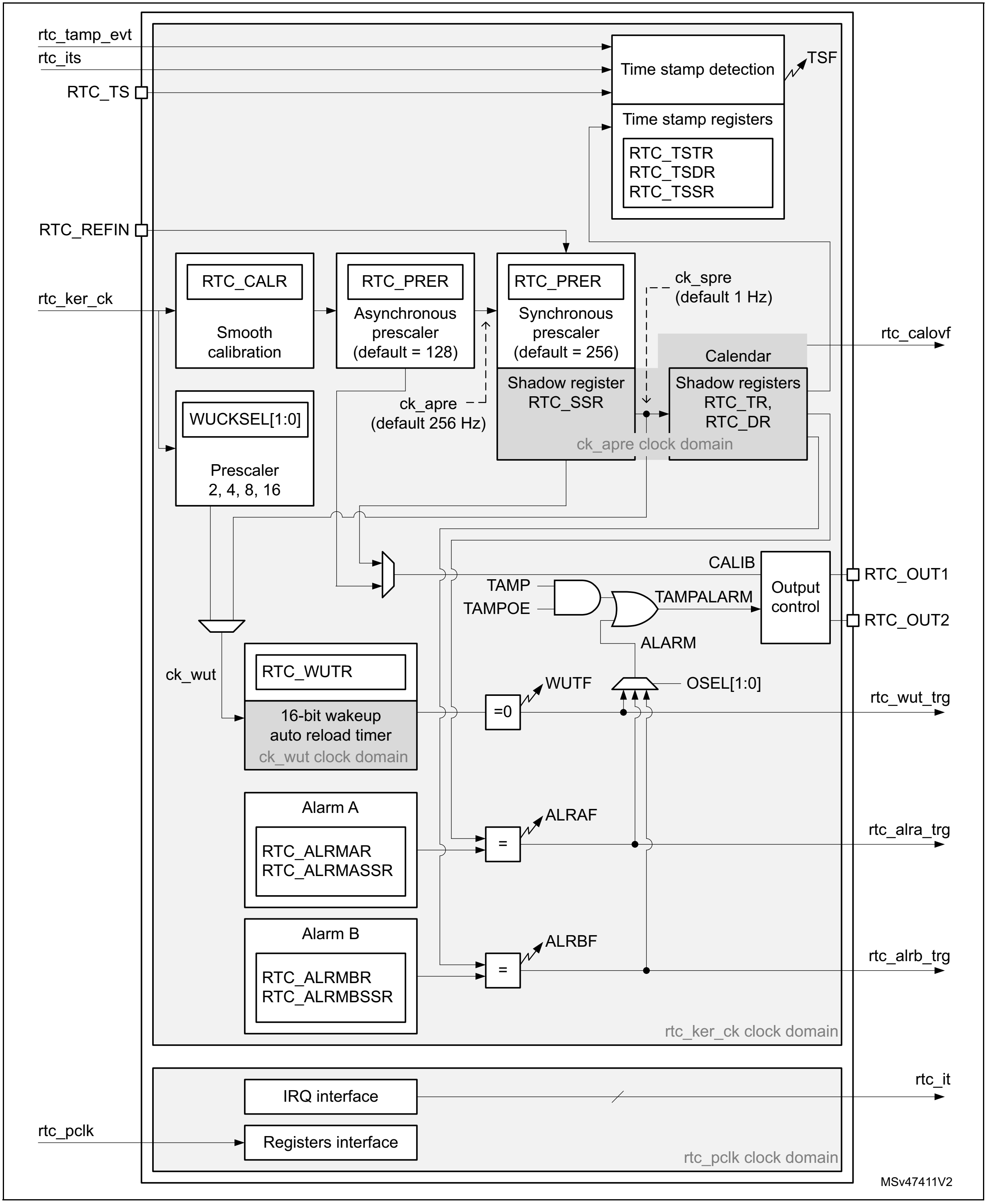
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE. When clocked by the LSI, the RTC is not functional in V_{BAT} mode, but is functional in all low-power modes except Shutdown mode.

All RTC events (Alarm, WakeUp Timer, Timestamp) can generate an interrupt and wakeup the device from the low-power modes.

35.3 RTC functional description

35.3.1 RTC block diagram

Figure 528. RTC block diagram



35.3.2 RTC pins and internal signals

Table 328. 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 which selects 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 329. 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 35.5: 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

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 35.4: RTC low-power modes* for more details.

Table 330. 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.

35.3.3 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 36: 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 331*.

Table 331. 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 331. 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 331. 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 PB2 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 show in table *Table 332*.

For PB2, the GPIO should be configured as an alternate function.

Table 332. RTC_OUT mapping

OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1 on PC13	RTC_OUT2 on PB2
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

35.3.4 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 *Section 7: Reset and clock control (RCC)*.

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 528: 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 35.3.7: Periodic auto-wakeup* for details).

35.3.5 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 35.6.10: 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.

35.3.6 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.

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.

35.3.7 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 to configure the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock)

  When ck_spre frequency is 1 Hz, this allows to achieve a wakeup time 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 1550*), 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).

The WUTF flag must then be cleared by software.

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.

35.3.8 RTC initialization and configuration

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.

Writing to the protected RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

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. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER 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 and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

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).
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. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[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.The WUTWF bit is cleared up to 2 RTCCLK clocks cycles after WUTE is cleared, due to clock synchronization.

35.3.9 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 cycles. 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 1549*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 35.3.11: 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.

35.3.10 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 and RTC_ALRMBSSR/RTC_ALRMBR).

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.

35.3.11 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.

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 can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (PREDIV_S + 1) seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this will delay the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

Caution: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

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: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

35.3.12 RTC reference clock detection

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.

35.3.13 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 RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK 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 RTCCLK 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 RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK 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 (FCAL) given the input frequency (FRTCCLK) is as follows:

$$F_{CAL} = F_{RTCCLK} \text{ x } [1 + (CALP \text{ x } 512 - CALM) / (2^{20} + CALM - CALP \text{ x } 512)]$$

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.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK 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:

$$F_{CAL} = F_{RTCCLK} \text{ x } [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

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

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.

35.3.14 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).

35.3.15 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 fRTCCLK/(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 6th stage of the asynchronous prescaler.

When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

35.3.16 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 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.

35.4 RTC low-power modes

Table 333. 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 334. 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

35.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

Table 335. 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]

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.

35.6 RTC registers

Refer to *Section 1.2 on page 73* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

35.6.1 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 1549* and *Reading the calendar on page 1551*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1549*.

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

35.6.2 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 1549* and *Reading the calendar on page 1551*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1549*.

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.

35.6.3 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
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
SS[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 **SS[15:0]**: Sub second value

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)

Note: 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.

35.6.4 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 1549*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx 000x xxxx (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.	Res.	Res.	Res.	Res.	Res.	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
								rw	r	rc_w0	r	r	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 15:8 Reserved, must be kept at reset value.

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). 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

Bit 1 **ALRBWF**: Alarm B write flag

This bit is set by hardware when alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC_CR.
It is cleared by hardware in initialization mode.
0: Alarm B update not allowed
1: Alarm B update allowed

Bit 0 **ALRAWF**: Alarm A write flag

This bit is set by hardware when alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.
It is cleared by hardware in initialization mode.
0: Alarm A update not allowed
1: Alarm A update allowed

35.6.5 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 1549*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1549*.

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)

35.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1549*.

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
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**
WUT[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 **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 + 1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

35.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1549.

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
OUT2 EN	TAMP ALRM_ TYPE	TAMP ALRM_ PU	Res.	Res.	TAMP OE	TAMP TS	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	ALRB IE	ALRA IE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYP SHAD	REFCK ON	TS EDGE	WUCKSEL[2:0]		
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 to remap the RTC outputs 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 35.3.15: 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

Note: When the wakeup timer is disabled, wait for WUTWF=1 before enabling it again.

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 Reserved, must be kept at reset value.

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: 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
11x: ck_spre (usually 1 Hz) clock is selected and 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.

35.6.8 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.

35.6.9 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1549*.

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	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **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 35.3.13: 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 35.3.13: 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 35.3.13: RTC smooth digital calibration.

Bits 12:9 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 35.3.13: RTC smooth digital calibration on page 1553*.

35.6.10 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1549*.

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))).

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.

35.6.11 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.

35.6.12 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

35.6.13 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
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
SS[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 SS[15:0]: Sub second value

SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

35.6.14 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1549*.

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.

35.6.15 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1549*.

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
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	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.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **MASKSS[3:0]**: 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[14:1] are don't care in alarm A comparison. Only SS[0] is compared.

2: SS[14:2] are don't care in alarm A comparison. Only SS[1:0] are compared.

3: SS[14:3] are don't care in alarm A comparison. Only SS[2:0] are compared.

...

12:SS[14:12] are don't care in alarm A comparison. SS[11:0] are compared.

13:SS[14:13] are don't care in alarm A comparison. SS[12:0] are compared.

14:SS[14] is don't care in alarm A comparison. SS[13:0] are compared.

15:All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Note: 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 A is to be activated. Only bits 0 up MASKSS-1 are compared.

35.6.16 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1549*.

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

35.6.17 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.

This register is write protected.The write access procedure is described in *Section : RTC register write protection*.

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
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	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.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0x0: 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).
0x1: SS[14:1] are don’t care in alarm B comparison. Only SS[0] is compared.
0x2: SS[14:2] are don’t care in alarm B comparison. Only SS[1:0] are compared.
0x3: SS[14:3] are don’t care in alarm B comparison. Only SS[2:0] are compared.
...
0xC: SS[14:12] are don’t care in alarm B comparison. SS[11:0] are compared.
0xD: SS[14:13] are don’t care in alarm B comparison. SS[12:0] are compared.
0xE: SS[14] is don’t care in alarm B comparison. SS[13:0] are compared.
0xF: All 15 SS bits are compared and must match to activate alarm.
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.

35.6.18 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.	Res.	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

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.

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.

35.6.19 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.	Res.	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

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.

35.6.20 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.	Res.	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
										w	w	w	w	w	w

Bits 31:6 Reserved, must be kept at reset value.

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.

35.6.21 RTC register map

Table 336. 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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	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.	Res.	Res.	Res.	Res.	Res.	INIT	INITF	RSF	INITS	SHPF	WUT WF	ALRBWF	ALRAWF
	Reset value																0									0	0	0	0	0	1	1	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUT[15:0]															
	Reset value																	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	Res.	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
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	Res.	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	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[1:0]			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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 336. 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**	Res.	Res.	Res.	Res.	MASKSS[3:0]				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
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**	Res.	Res.	Res.	Res.	MASKSS[3:0]				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
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.	Res.	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																											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.	Res.	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																											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.	Res.	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																											0	0	0	0	0	0

Refer to *Section 2.2 on page 81* for the register boundary addresses.

36 Tamper and backup registers (TAMP)

36.1 Introduction

32 (category 3 and category 4 devices) or 16 (category 2 devices) 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.

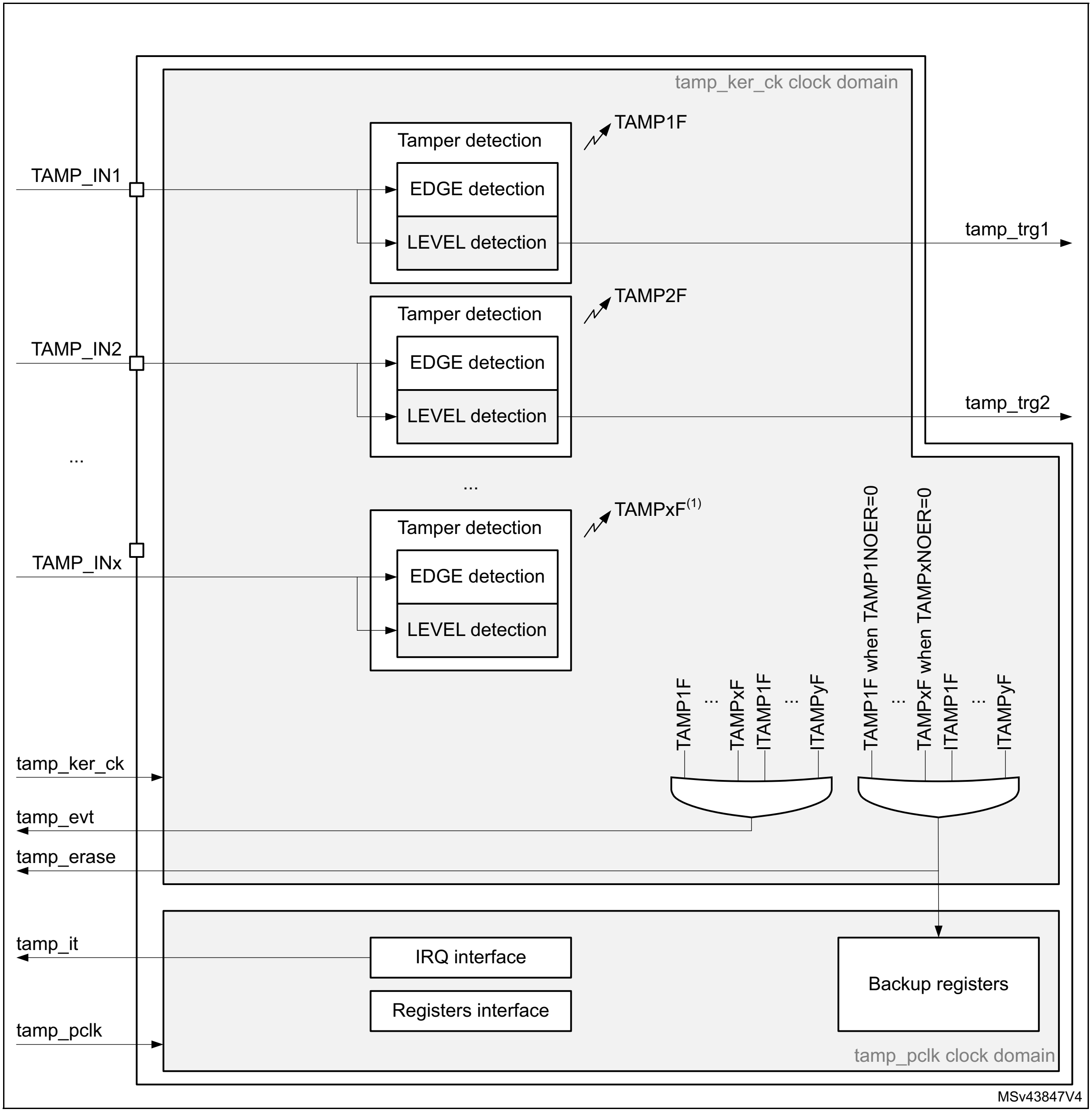
36.2 TAMP main features

- 32 (category 3 and category 4 devices) or 16 (category 2 devices) 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.

36.3 TAMP functional description

36.3.1 TAMP block diagram

Figure 529. TAMP block diagram



1. The number of external and internal tampers depends on products.

36.3.2 TAMP pins and internal signals

Table 337. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin

Table 338. 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 tamper event detection (internal or external)
tamp_it	Output	TAMP interrupt (refer to *Section 36.5: 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 36.4: TAMP low-power modes* for more details.

Table 339. 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
tamp_itamp3	LSE monitoring
tamp_itamp4	HSE monitoring
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	ST manufacturer readout

36.3.3 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.

36.3.4 Tamper detection

The tamper detection can be configured for the following purposes:

- erase the backup registers (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.

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 to trig automatically the low-power timers in Stop mode, 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 (TAMPFLT ≠ 00 and active mode is not selected).

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 datasheet for the electrical characteristics of the pull-up resistors.

36.4 TAMP low-power modes

Table 340. 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.

36.5 TAMP interrupts

The interrupt channel is set in the interrupt status register. The interrupt output is also activated.

Table 341. 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)
	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.

36.6 TAMP registers

Refer to *Section 1.2 on page 73* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

36.6.1 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.	Res.	Res.	ITAMP6E	ITAMP5E	ITAMP4E	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 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6E**: Internal tamper 6 enable: ST manufacturer readout

0: Internal tamper 6 disabled.

1: Internal tamper 6 enabled: a tamper is generated in case of ST manufacturer readout.

Bit 20 **ITAMP5E**: Internal tamper 5 enable: RTC calendar overflow

0: Internal tamper 5 disabled.

1: Internal tamper 5 enabled: a tamper 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.

Bit 19 **ITAMP4E**: Internal tamper 4 enable: HSE monitoring

0: Internal tamper 4 disabled.

1: Internal tamper 4 enabled. a tamper is generated when the HSE frequency is below or above thresholds.

Bit 18 **ITAMP3E**: Internal tamper 3 enable: LSE monitoring

0: Internal tamper 3 disabled.

1: Internal tamper 3 enabled: a tamper is generated when the LSE frequency is below or above thresholds.

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

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

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

0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

36.6.2 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	Res.	Res.	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
					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.	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 (active mode disabled)

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 (active mode disabled)

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 (active mode disabled)

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 Reserved, must be kept at reset value.

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 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 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 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.

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.

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.

36.6.3 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.

36.6.4 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.	Res.	Res.	ITAMP6 IE	ITAMP5 IE	ITAMP4 IE	ITAMP3 IE	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.	TAMP 3IE	TAMP 2IE	TAMP 1IE
													rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable: ST manufacturer readout
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable: RTC calendar overflow
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable: HSE monitoring
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable: LSE monitoring
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.

36.6.5 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.	Res.	Res.	ITAMP6F	ITAMP5F	ITAMP4F	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 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6F**: ST manufacturer readout tamper detection flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: RTC calendar overflow tamper detection flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: HSE monitoring tamper detection flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: LSE monitoring tamper detection 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.

36.6.6 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.	Res.	Res.	ITAMP6 MF	ITAMP5 MF	ITAMP4 MF	ITAMP3 MF	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.	TAMP 3MF	TAMP 2MF	TAMP 1MF
													r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6MF**: ST manufacturer readout tamper interrupt masked flag

This flag is set by hardware when the internal tamper 6 interrupt is raised.

Bit 20 **ITAMP5MF**: RTC calendar overflow tamper interrupt masked flag

This flag is set by hardware when the internal tamper 5 interrupt is raised.

Bit 19 **ITAMP4MF**: HSE monitoring tamper interrupt masked flag

This flag is set by hardware when the internal tamper 4 interrupt is raised.

Bit 18 **ITAMP3MF**: LSE monitoring tamper 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.

36.6.7 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.	Res.	Res.	C ITAMP 6F	C ITAMP 5F	C ITAMP 4F	C ITAMP 3F	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.	CTAMP 3F	CTAMP 2F	CTAMP 1F
													w	w	w

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

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 **CITAMP4F**: Clear ITAMP4 detection flag

Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

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.

36.6.8 TAMP backup x register (TAMP_BKPxR)

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

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.

36.6.9 TAMP register map

Table 342. 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.	Res.	Res.	ITAMP6E	ITAMP5E	ITAMP4E	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	Res.	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
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.	Res.	Res.	ITAMP6IE	ITAMP5IE	ITAMP4IE	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.	Res.	Res.	ITAMP6F	ITAMP5F	ITAMP4F	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.	Res.	Res.	ITAMP6MF	ITAMP5MF	ITAMP4MF	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.	Res.	Res.	CITAMP6F	CITAMP5F	CITAMP4F	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
0x100 + 0x04*x, (x = 0 to 31)	**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 2.2 on page 81* for the register boundary addresses.

37 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

37.1 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.

37.2 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

37.3 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

37.4 USART implementation

The table below describes USART implementation on STM32G4 Series devices. It also includes LPUART for comparison.

Table 343. USART / LPUART features

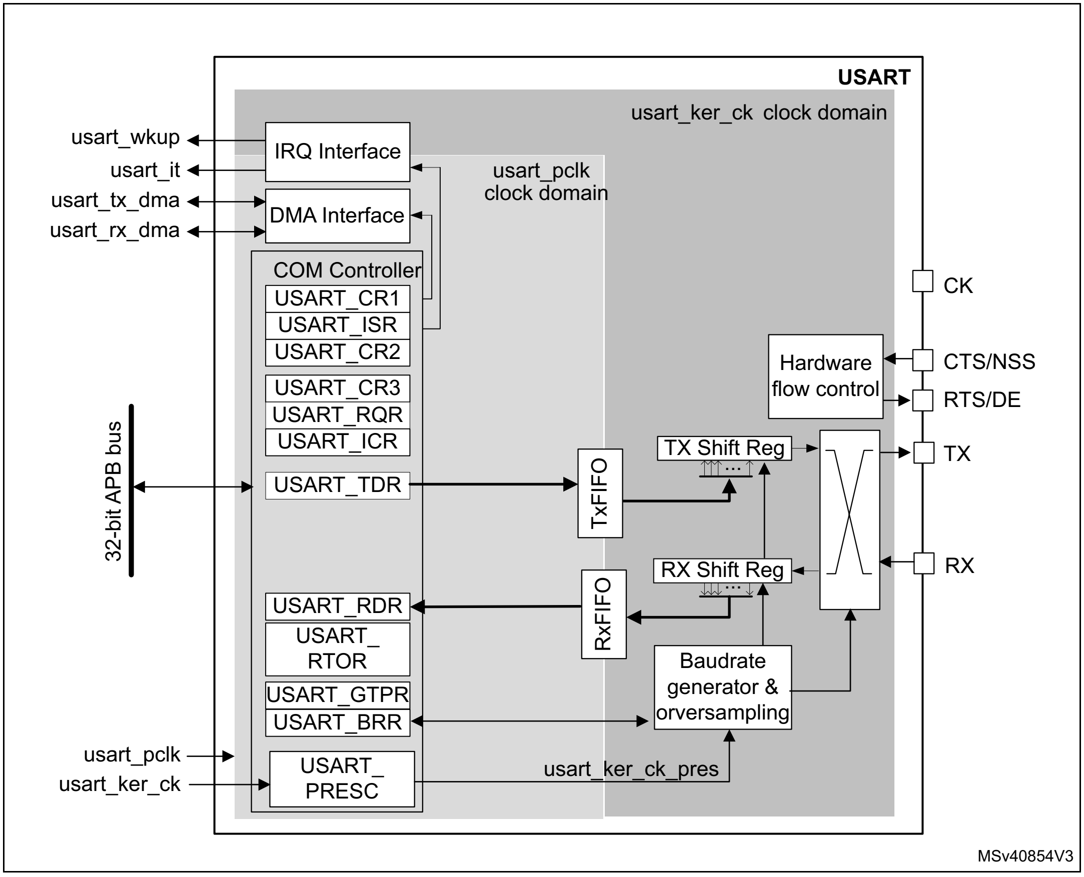
USART / LPUART modes/features[1]	USART1/2/3	UART4/5	LPUART1
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	-	-
Smartcard mode	X	-	-
Single-wire Half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain and wakeup from low-power mode	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver Enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size	8		

1. X = supported.

37.5 USART functional description

37.5.1 USART block diagram

Figure 530. USART block diagram



The simplified block diagram given in *Figure 530* 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 SCLK 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.

37.5.2 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.

37.5.3 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 531*):

- 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.

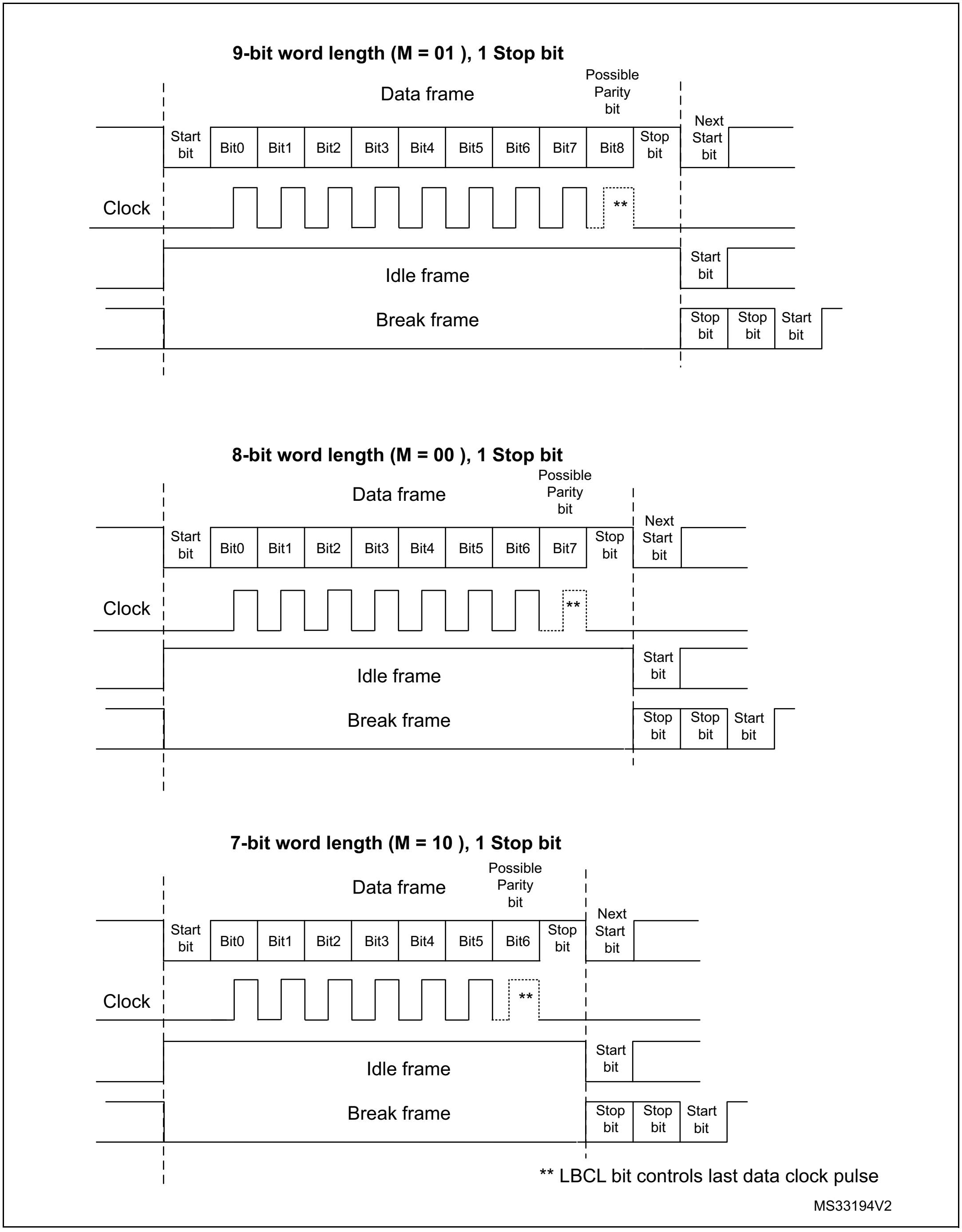
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

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 531. Word length programming



37.5.4 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.

37.5.5 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 SCLK 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

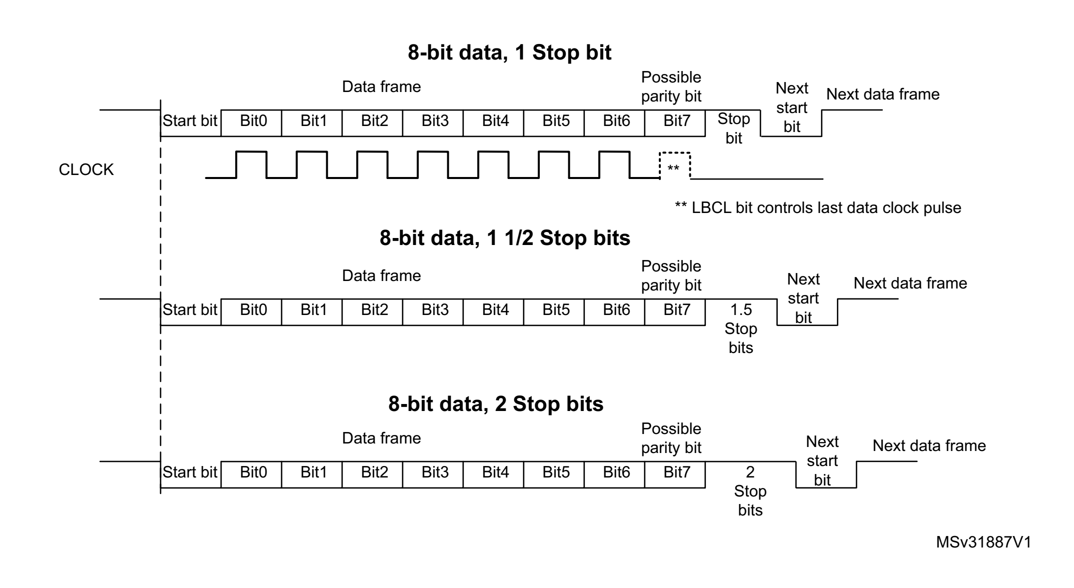
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***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 532*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 532. 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 37.5.10: 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.

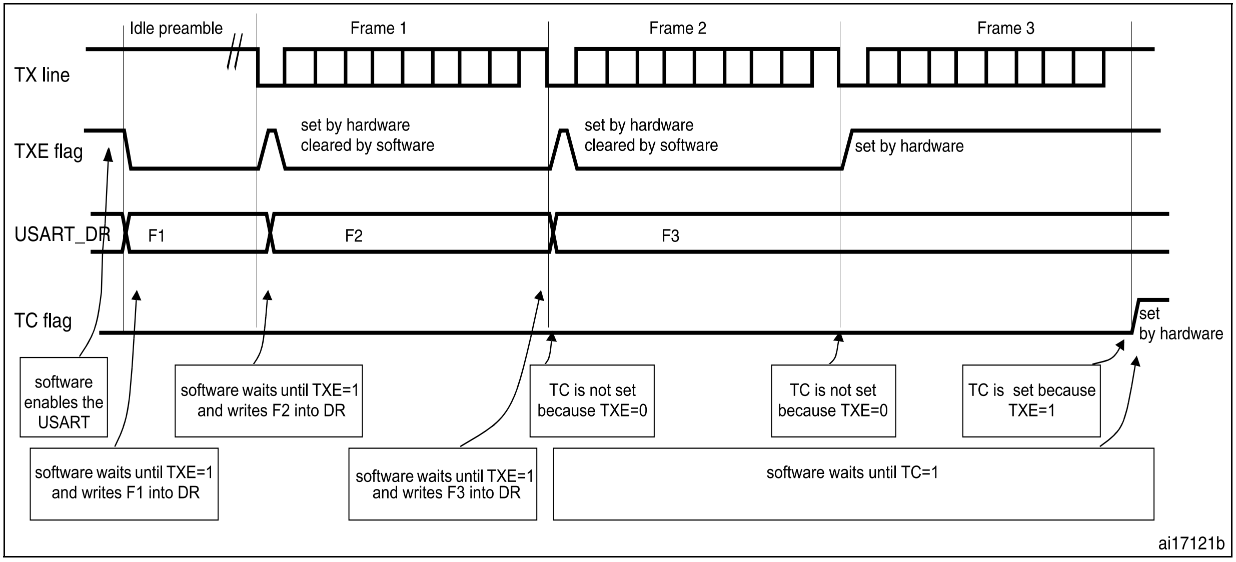
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  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 533: TC/TXE behavior when transmitting*).

Figure 533. TC/TXE behavior when transmitting


Idle preamble
Frame 1
Frame 2
Frame 3
TX line
TXE flag
set by hardware
cleared by software
set by hardware
cleared by software
set by hardware
USART_DR
F1
F2
F3
TC flag
set
by hardware
software enables the USART
software waits until TXE=1 and writes F2 into DR
TC is not set because TXE=0
TC is not set because TXE=0
TC is set because TXE=1
software waits until TXE=1 and writes F1 into DR
software waits until TXE=1 and writes F3 into DR
software waits until TC=1
ai17121b

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 531*).

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.

37.5.6 USART receiver

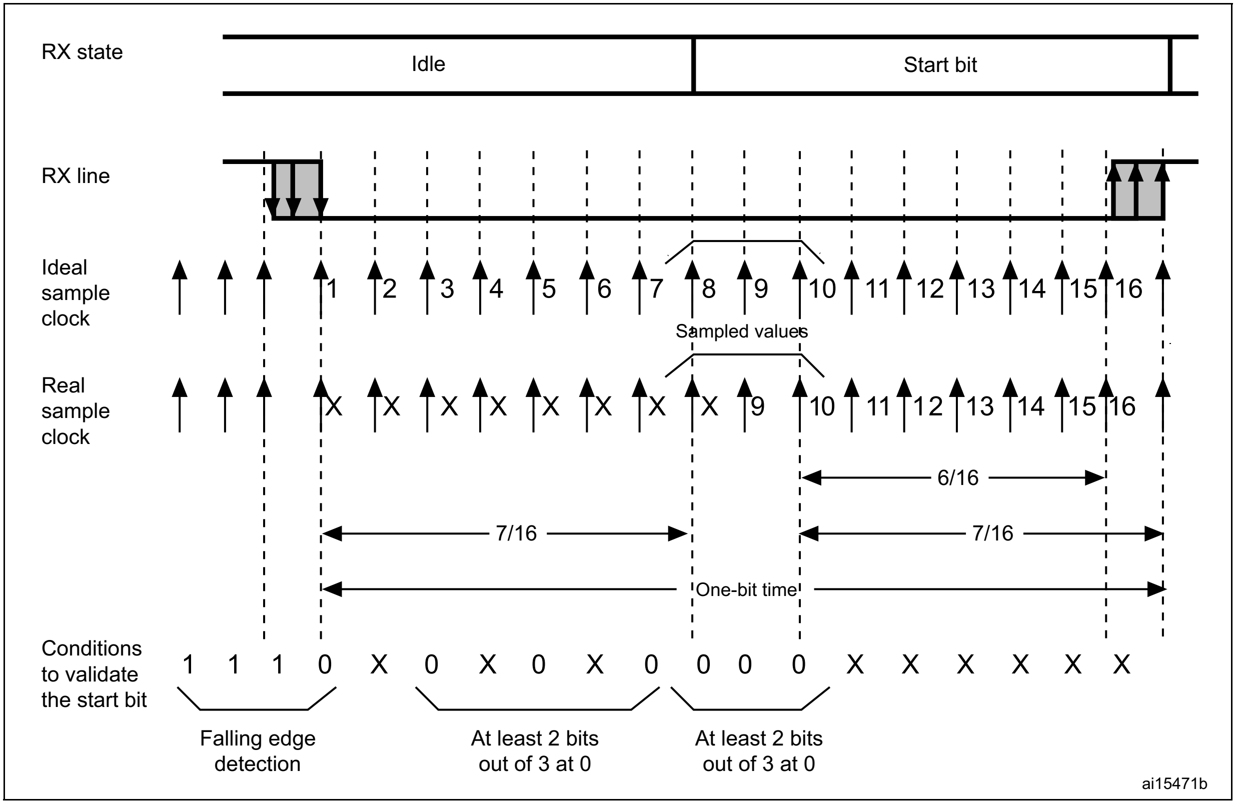
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

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 534. Start bit detection when oversampling by 16 or 8


RX state
Idle
Start bit
RX line
Ideal sample clock
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Sampled values
Real sample clock
X X X X X X X X 9 10 11 12 13 14 15 16
6/16
7/16
7/16
One-bit time
Conditions to validate the start bit
1 1 1 0 X 0 X 0 X 0 0 0 0 X X X X X X
Falling edge detection
At least 2 bits out of 3 at 0
At least 2 bits out of 3 at 0
ai15471b

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 37.5.10: 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:

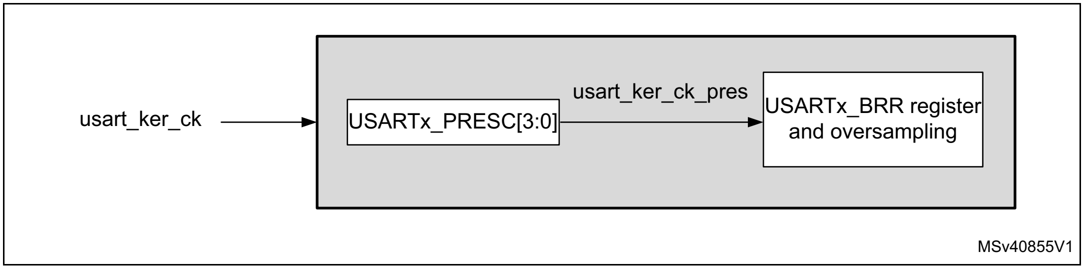
- Possible use of the USART in low-power mode
- Communication speed.

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 535. 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 536* and *Figure 537*).

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 37.5.8: Tolerance of the USART receiver to clock deviation on page 1614*)
- 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 344*) 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 37.5.8: Tolerance of the USART receiver to clock deviation on page 1614*). 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 536. Data sampling when oversampling by 16

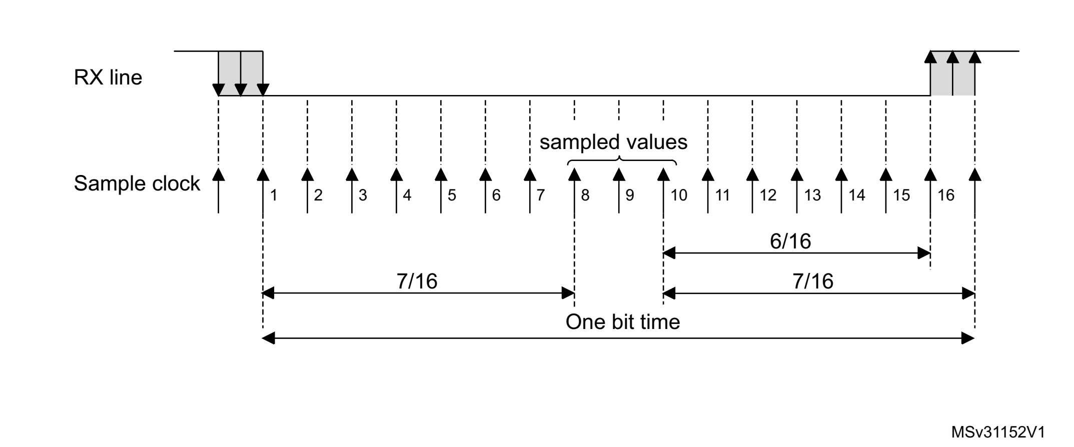


Figure 537. Data sampling when oversampling by 8

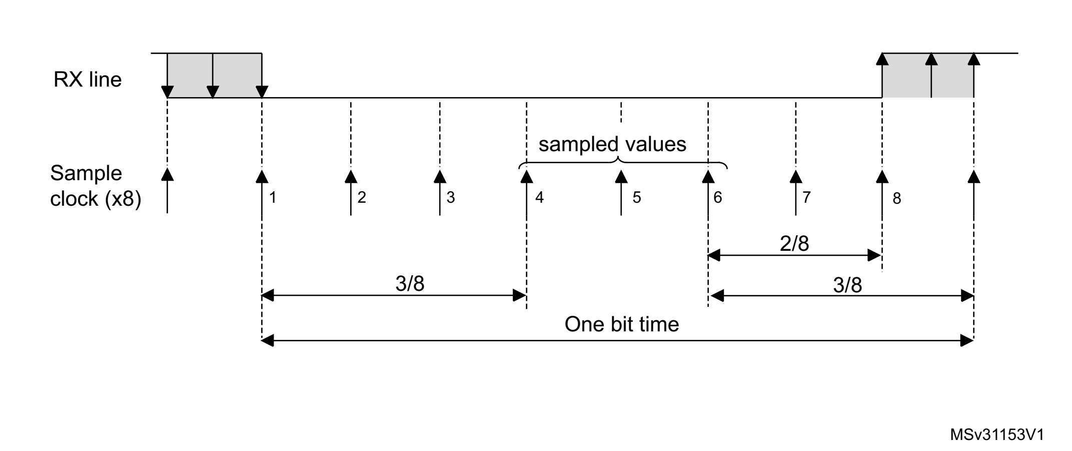


Table 344. Noise detection from sampled data

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 37.5.16: USART receiver timeout on page 1628* 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.

37.5.7 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

37.5.8 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)

$$\mathrm{DTRA} + \mathrm{DQUANT} + \mathrm{DREC} + \mathrm{DTCL} + \mathrm{DWU} < \text{USART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

when M[1:0] = 01:

$$\mathrm{DWU} = \frac{t_{\mathrm{WUUSART}}}{11 \times \mathrm{Tbit}}$$

when M[1:0] = 00:

$$\mathrm{DWU} = \frac{t_{\mathrm{WUUSART}}}{10 \times \mathrm{Tbit}}$$

when M[1:0] = 10:

$$\mathrm{DWU} = \frac{t_{\mathrm{WUUSART}}}{9 \times \mathrm{Tbit}}$$

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 345*, *Table 346*, 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 345. 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 346. 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 345 and *Table 346 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).*

37.5.9 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.

37.5.10 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:

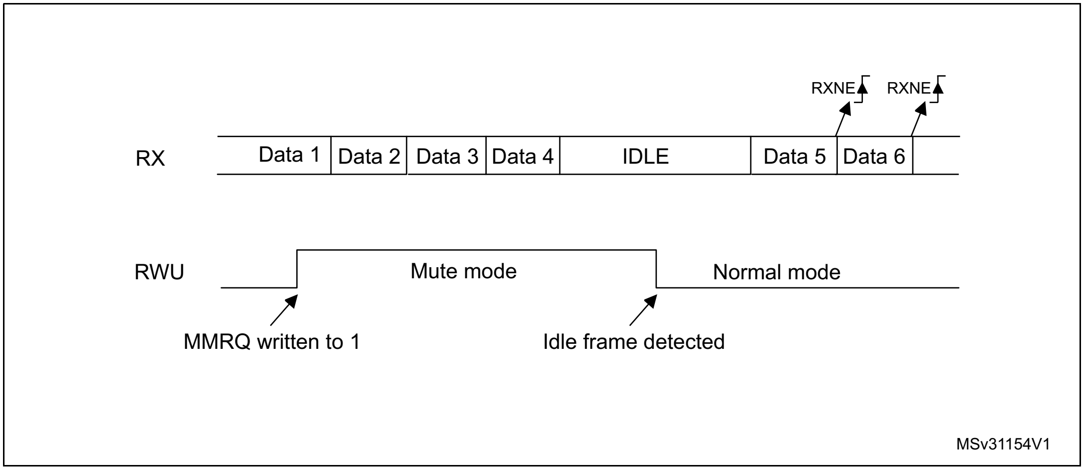
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

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 538*.

Figure 538. 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.

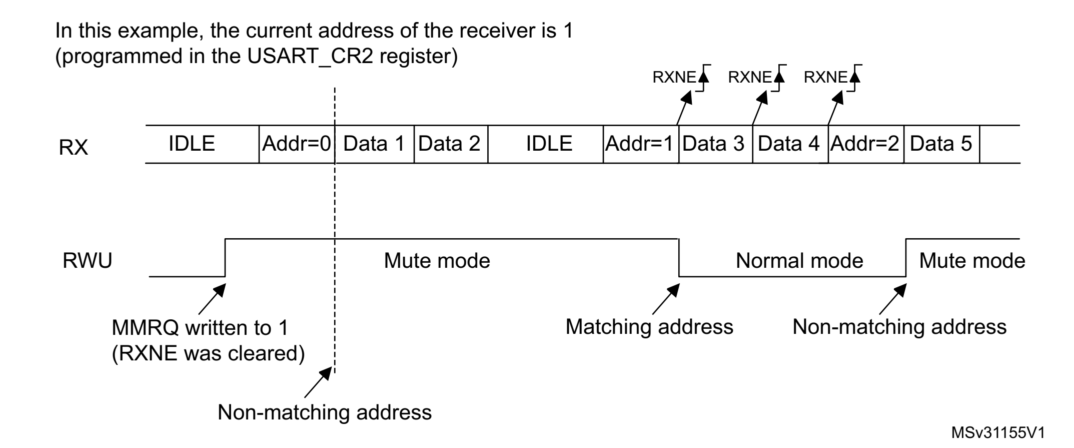
The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 539*.

Figure 539. Mute mode using address mark detection



37.5.11 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.

37.5.12 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 347*.

Table 347. 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).

37.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 37.4: USART implementation on page 1597*.

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 37.5.4* 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 540: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 1623*.

Examples of break frames are given on *Figure 541: Break detection in LIN mode vs. Framing error detection on page 1624*.

Figure 540. Break detection in LIN mode (11-bit break length - LBDL bit is set)

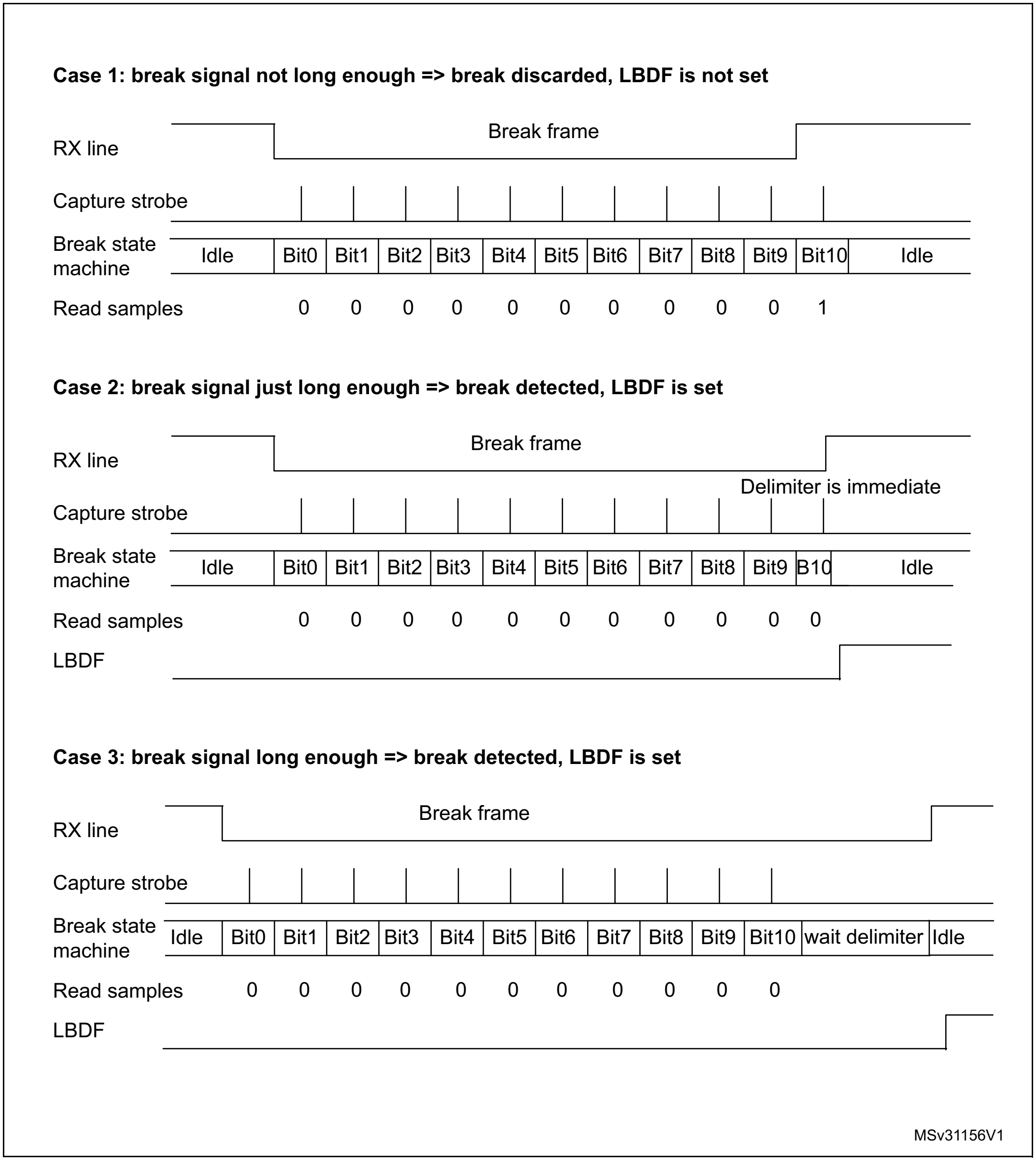
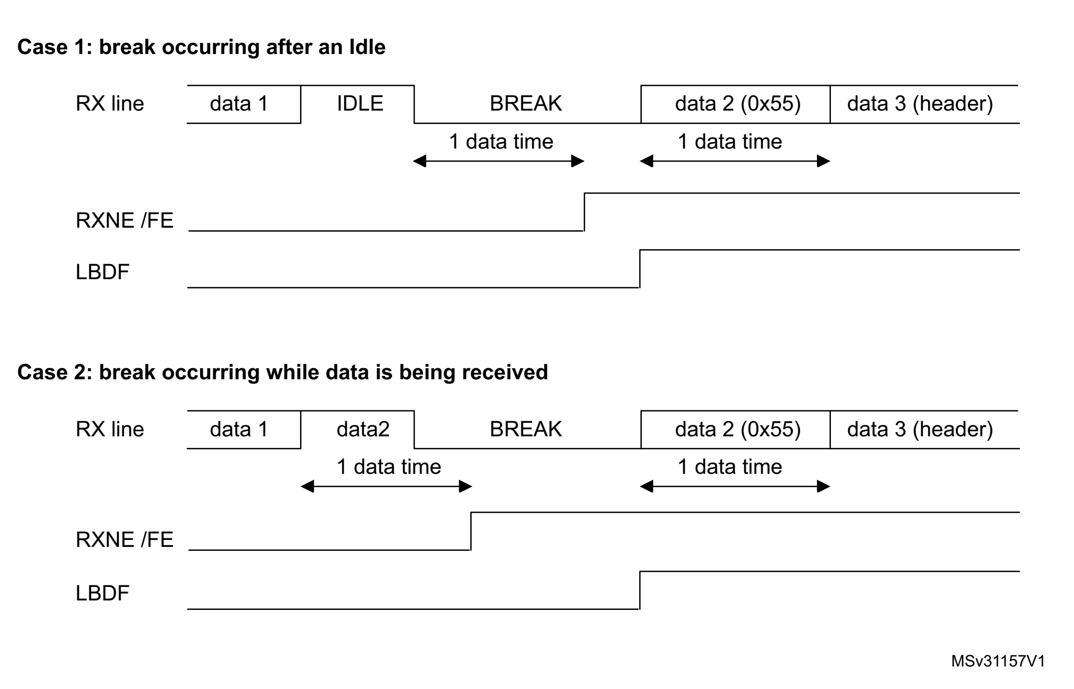


Figure 541. Break detection in LIN mode vs. Framing error detection



37.5.14 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 SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK 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 542*, *Figure 543* and *Figure 544*).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since SCLK 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 SCLK (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 SCLK 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 542. USART example of synchronous master transmission

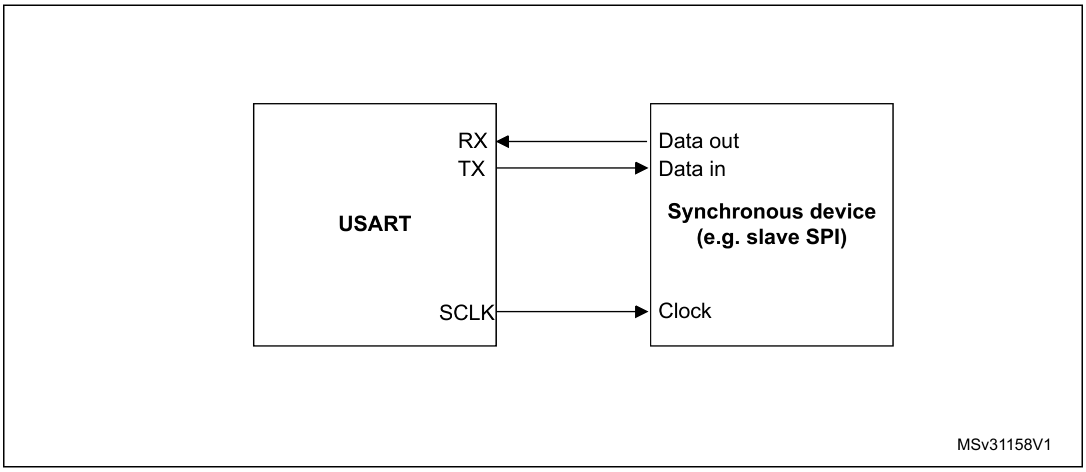


Figure 543. USART data clock timing diagram in synchronous master mode (M bits = 00)

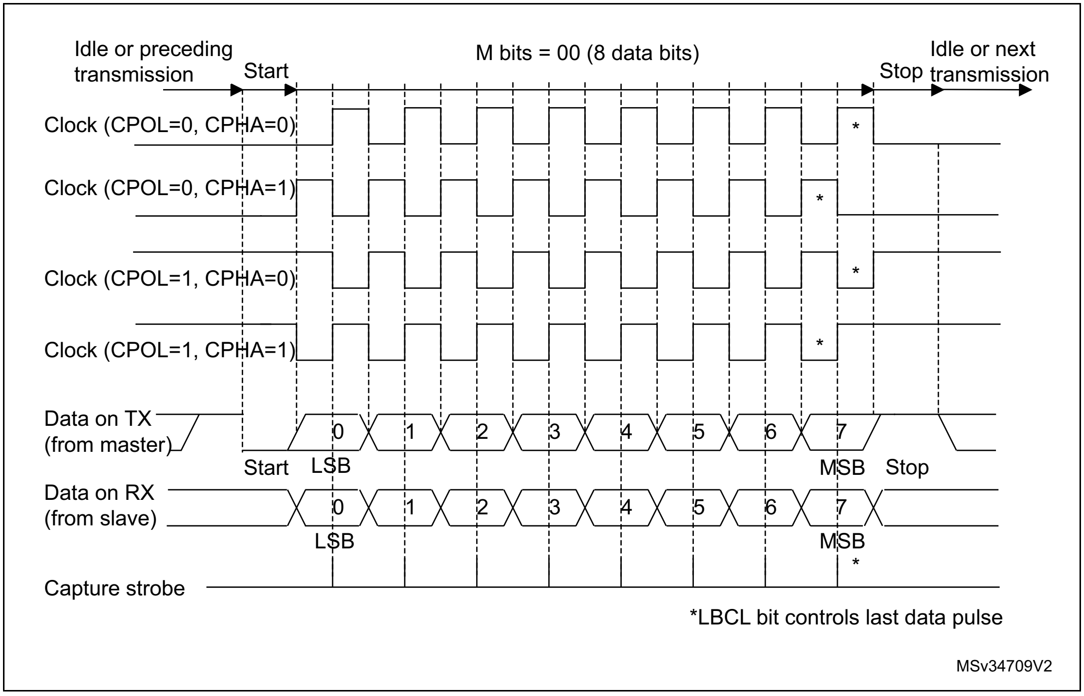
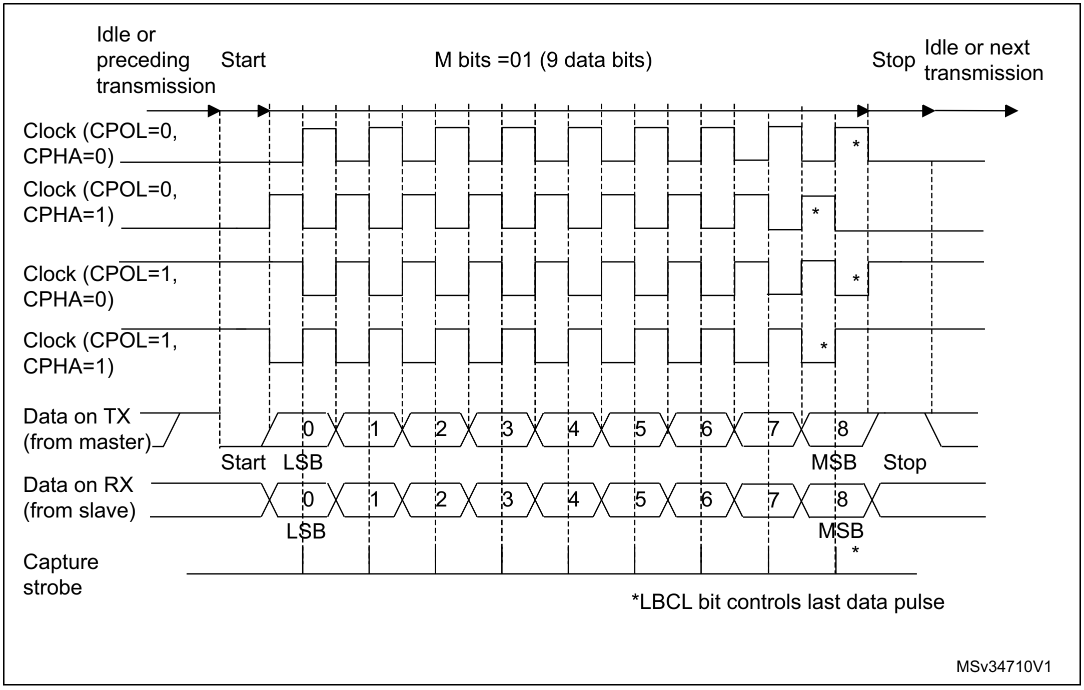


Figure 544. USART data clock timing diagram in synchronous master mode (M bits = 01)



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 SCLK pin is the input of the USART in slave mode.

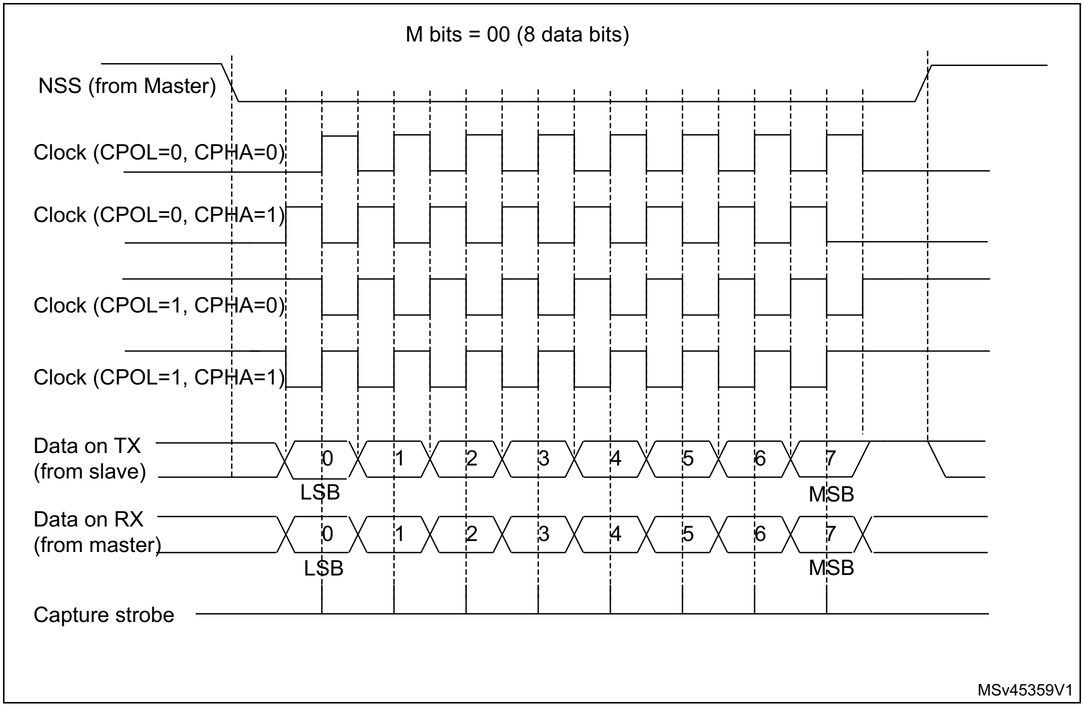
Note: *When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

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 545*).

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 545. 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 SCLK transmission edge. To avoid this underrun error, the USART_TDR should be written 3 usart_ker_ck *cycles before the first SCLK edge.*

37.5.15 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.

37.5.16 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.

37.5.17 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to *Section 37.4: USART implementation on page 1597*.

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.

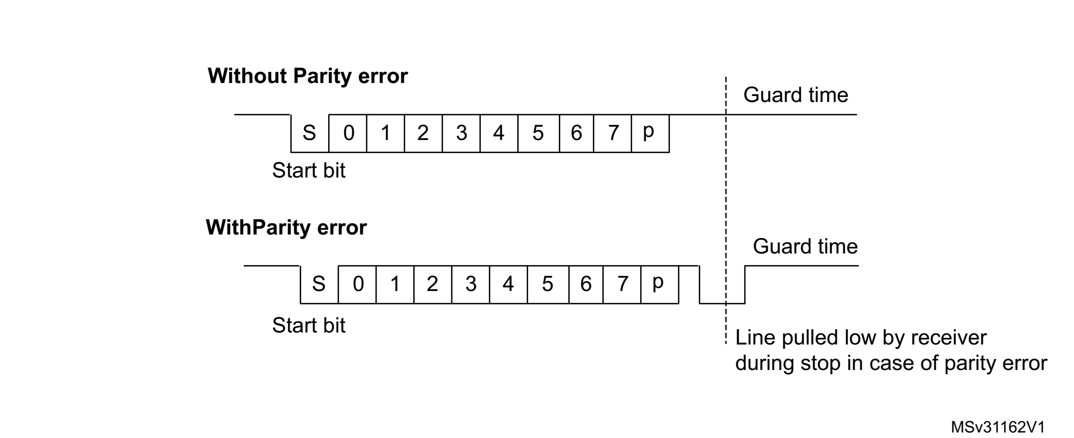
The USART should be configured as:

- 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 546 shows examples of what can be seen on the data line with and without parity error.

Figure 546. 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.

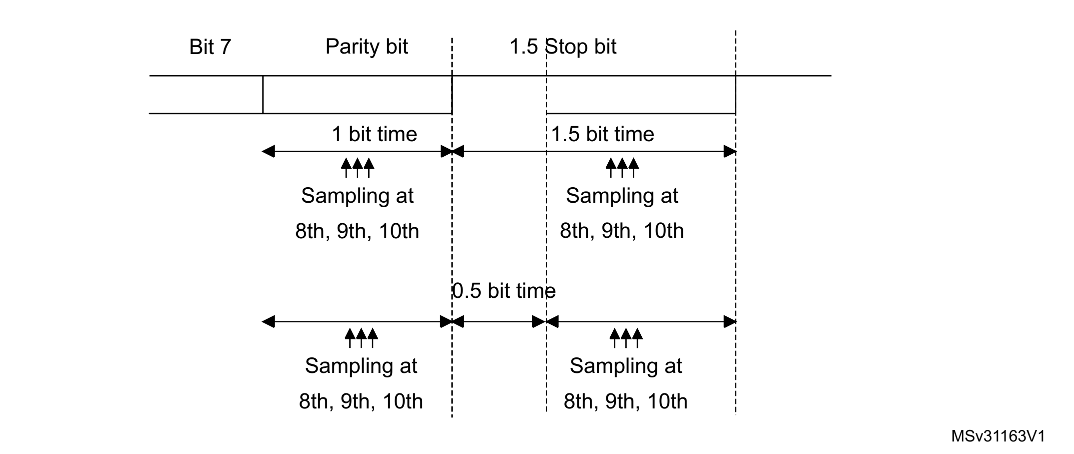
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it 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 547 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 547. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the Smartcard through the SCLK output. In Smartcard mode, SCLK 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. SCLK 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.

37.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 37.4: USART implementation on page 1597*.

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 548*).

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 Kbps 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 549*).
- 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 548. IrDA SIR ENDEC block diagram

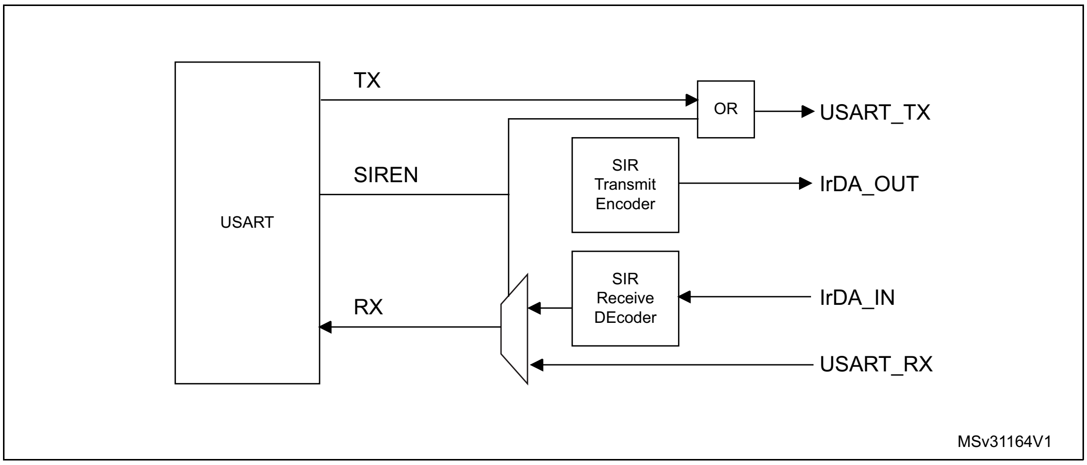
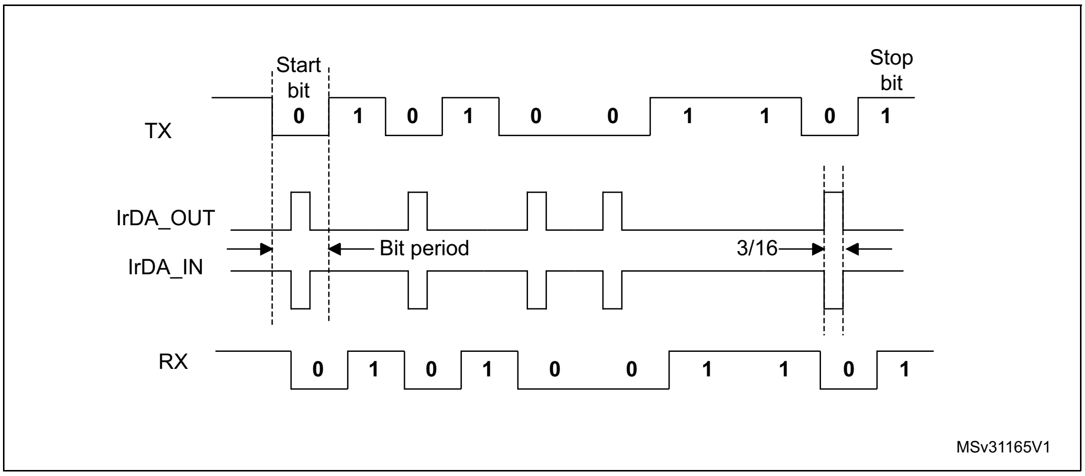


Figure 549. IrDA data modulation (3/16) - Normal mode



37.5.19 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 37.4: USART implementation on page 1597 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 37.5.6. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

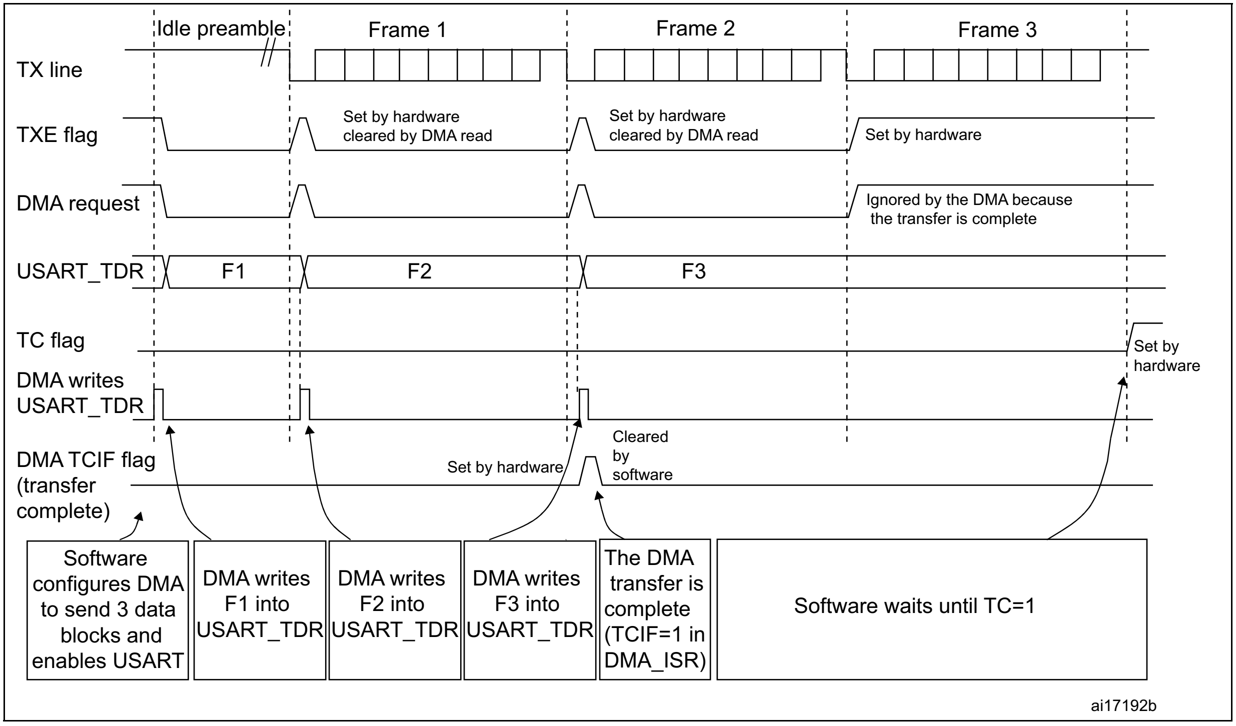
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode 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 550. Transmission using DMA


Idle preamble
Frame 1
Frame 2
Frame 3
TX line
TXE flag
Set by hardware cleared by DMA read
Set by hardware cleared by DMA read
Set by hardware
DMA request
Ignored by the DMA because the transfer is complete
USART_TDR
F1
F2
F3
TC flag
Set by hardware
DMA writes USART_TDR
DMA TCIF flag (transfer complete)
Set by hardware
Cleared by software
Software configures DMA to send 3 data blocks and enables USART
DMA writes F1 into USART_TDR
DMA writes F2 into USART_TDR
DMA writes F3 into USART_TDR
The DMA transfer is complete (TCIF=1 in DMA_ISR)
Software waits until TC=1
ai17192b

Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

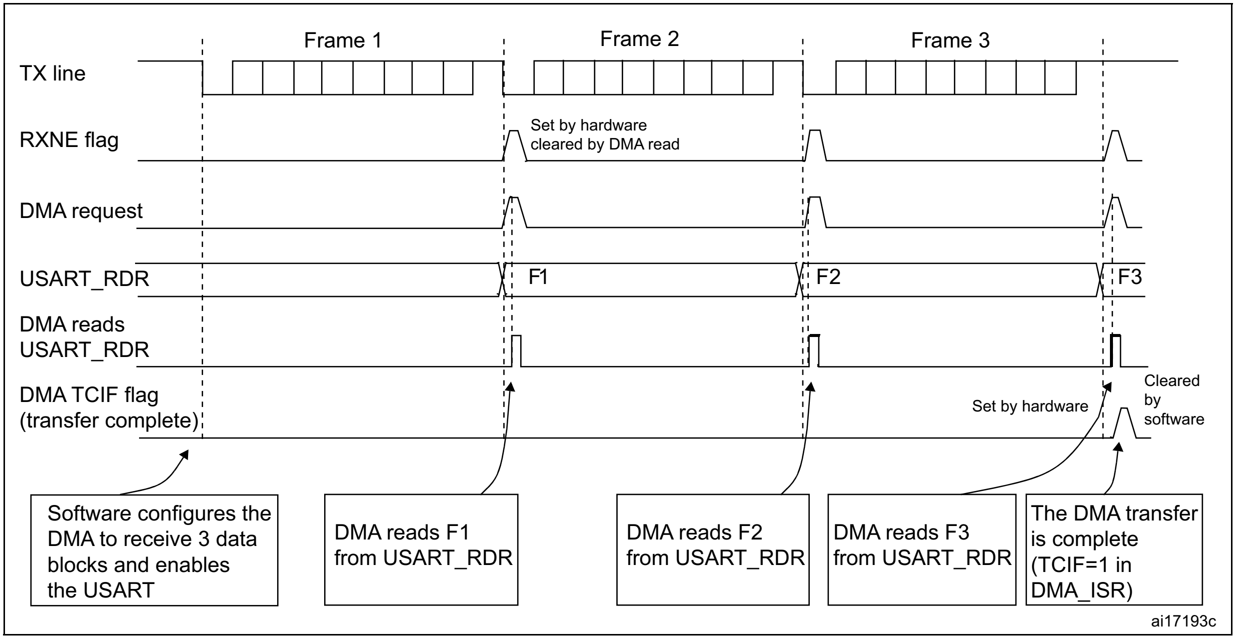
Reception using DMA

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 551. Reception using DMA


Frame 1
Frame 2
Frame 3
TX line
RXNE flag
Set by hardware
cleared by DMA read
DMA request
USART_RDR
F1
F2
F3
DMA reads
USART_RDR
DMA TCIF flag
(transfer complete)
Set by hardware
Cleared
by
software
Software configures the DMA to receive 3 data blocks and enables the USART
DMA reads F1 from USART_RDR
DMA reads F2 from USART_RDR
DMA reads F3 from USART_RDR
The DMA transfer is complete (TCIF=1 in DMA_ISR)
ai17193c

Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

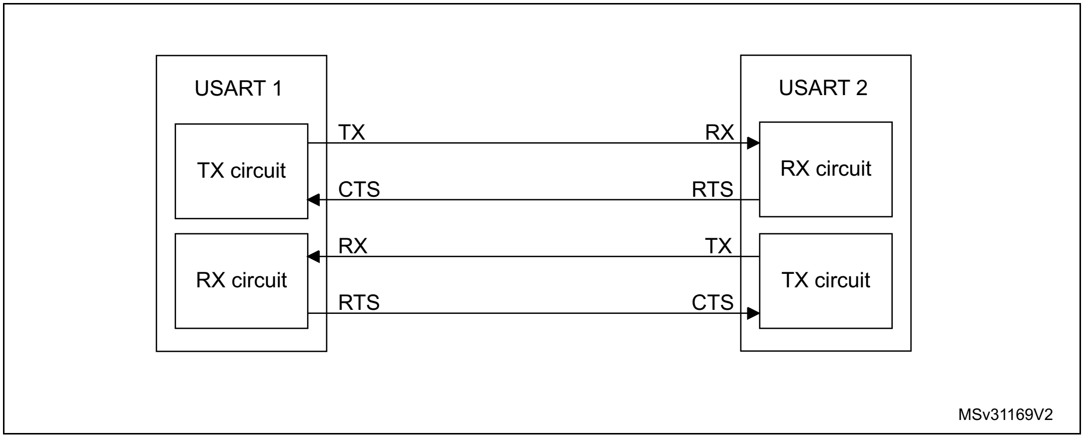
Error flagging and interrupt generation in multibuffer communication

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.

37.5.20 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The *Figure 552* shows how to connect 2 devices in this mode:

Figure 552. Hardware flow control between 2 USARTs

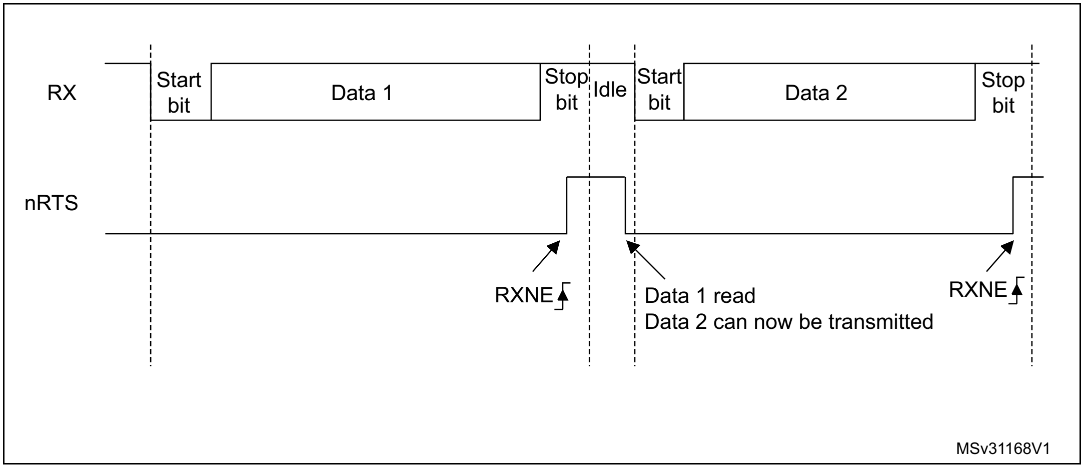


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to ‘1’ in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 553* shows an example of communication with RTS flow control enabled.

Figure 553. RS232 RTS flow control



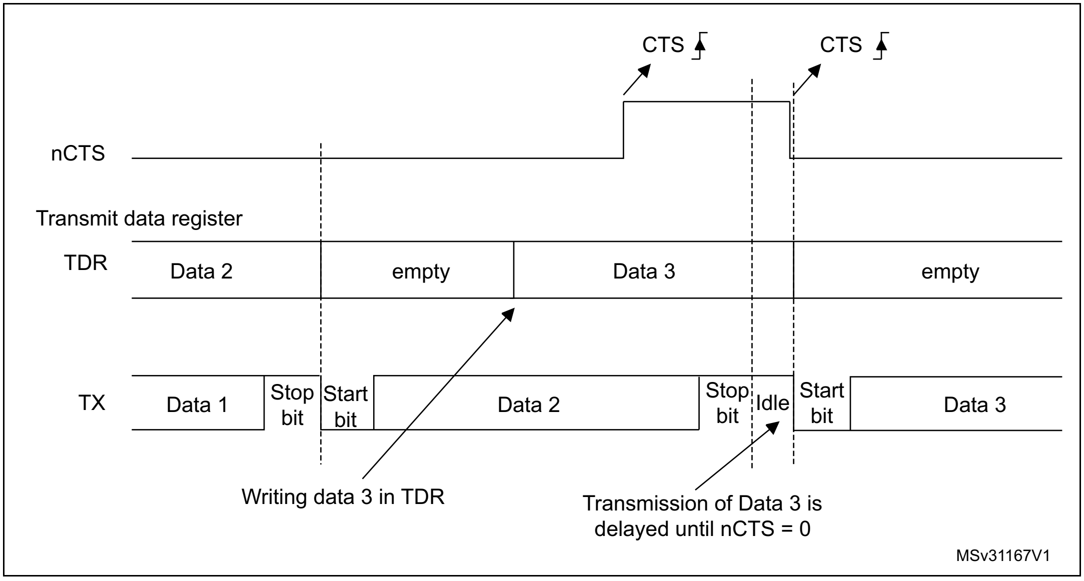
Note: When FIFO mode is enabled, nRTS is deasserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (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 nCTS is deasserted 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 nCTS 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 554* shows an example of communication with CTS flow control enabled.

Figure 554. RS232 CTS flow control



Note: For correct behavior, nCTS must be asserted 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).

37.5.21 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. In this case the **usart_wkup** interrupt is not mandatory and setting the WUF being is sufficient to wake up the MCU from low-power mode.

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 555 shows the USART behavior when the wakeup event is verified.

Figure 555. Wakeup event verified (wakeup event = address match, FIFO disabled)

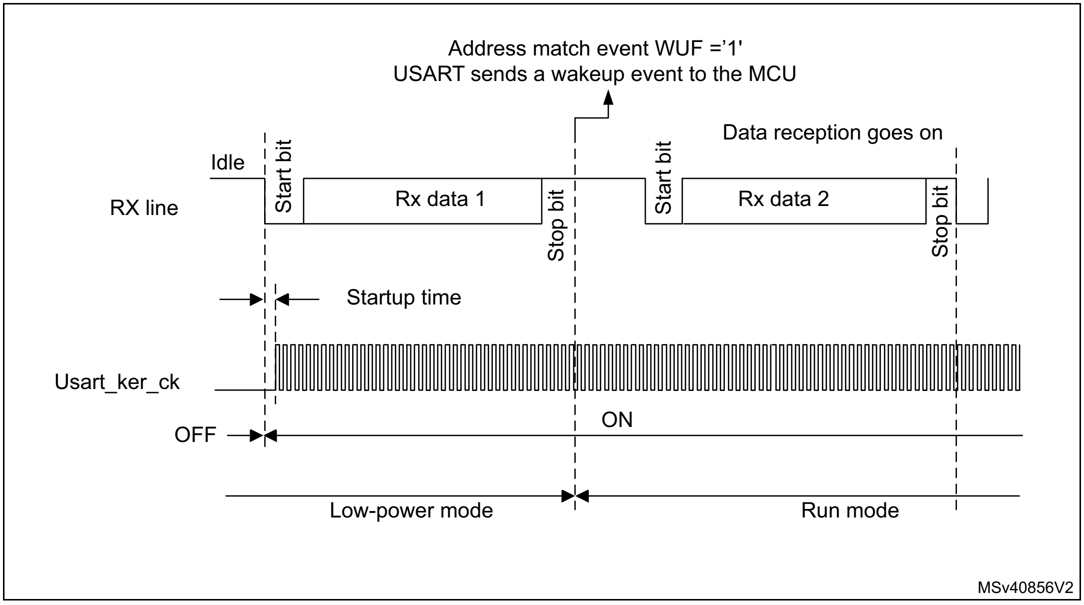
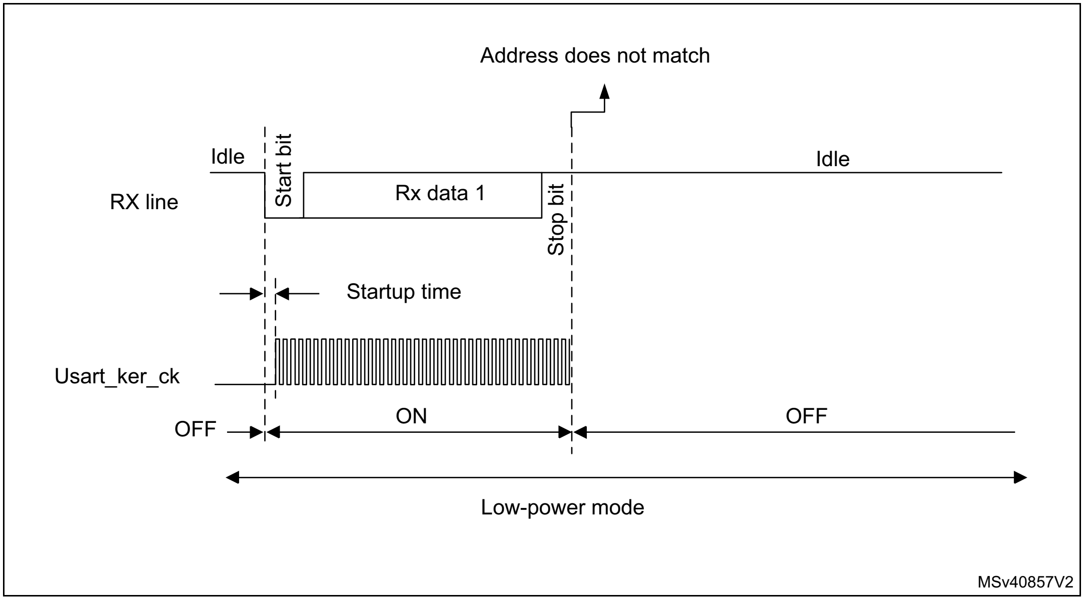


Figure 556 shows the USART behavior when the wakeup event is not verified.

Figure 556. 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 37.5.8: 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 345: 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 HSI16 is used as usart_ker_ck, and the HSI16 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.

37.6 USART in low-power modes

Table 348. 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. 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 37.4: 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.

37.7 USART interrupts

Refer to *Table 349* for a detailed description of all USART interrupt requests.

Table 349. 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 37.4: 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.

37.8 USART registers

Refer to *Section 1.2 on page 73* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

37.8.1 USART control register 1 [alternate] (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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 SCLK is always available when CLKEN = 1, regardless of the UE bit value.

37.8.2 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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 SCLK is always available when CLKEN = 1, regardless of the UE bit value.

37.8.3 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

ADD[7:4]:

These bits give the address of the USART node or a character code to be recognized.

They are used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or low-power mode. The MSB of the character sent by the transmitter should be equal to 1. They can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.

These bits can only be written when reception is disabled (RE = 0) or the USART is disabled (UE = 0).

ADD[3:0]:

These bits give the address of the USART node or a character code to be recognized.

They are used for wakeup with address mark detection, in multiprocessor communication during Mute mode or low-power mode.

These bits can only be written when reception is disabled (RE = 0) or 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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 SCLK pin.

0: SCLK pin disabled

1: SCLK 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 37.4: USART implementation on page 1597.

In Smartcard mode, in order to provide correctly the SCLK 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 SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on SCLK pin outside transmission window

1: Steady high value on SCLK 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 37.4: USART implementation on page 1597.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 536* and *Figure 537*)

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 37.4: USART implementation on page 1597.

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 SCLK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the SCLK pin

1: The clock pulse of the last data bit is output to the SCLK 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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

37.8.4 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]			RXFTIE	RXFTCFG[2:0]			TCBGTIE	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	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

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 37.4: USART implementation on page 1597.

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 nRTS output is asserted (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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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.

37.8.5 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.

37.8.6 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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

37.8.7 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 37.4: USART implementation on page 1597.

37.8.8 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 37.4: USART implementation on page 1597.

37.8.9 USART interrupt and status register [alternate] (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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 nCTS input pin.

0: nCTS line set

1: nCTS 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 nCTS 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 nCTS status line

1: A change occurred on the nCTS 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 37.4: USART implementation on page 1597.

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 or EIE = 1 in the USART_CR1 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 37.5.8: Tolerance of the USART receiver to clock deviation on page 1614).

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_CR1 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.

37.8.10 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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 nCTS input pin.

0: nCTS line set

1: nCTS 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 nCTS 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 nCTS status line

1: A change occurred on the nCTS 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 37.4: USART implementation on page 1597.

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 USART_CR1 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 37.5.8: Tolerance of the USART receiver to clock deviation on page 1614).

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_CR1 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

37.8.11 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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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 37.4: USART implementation on page 1597.

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.

37.8.12 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 530*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

37.8.13 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 530*).

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.

37.8.14 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.

37.8.15 USART register map

The table below gives the USART register map and reset values.

Table 350. 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

Table 350. 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
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
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 2.2: Memory organization* for the register boundary addresses.

38 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

38.1 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/s. 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.

38.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud/s to 9600 baud/s 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

38.3 LPUART implementation

Below the description of LPUART implementation in comparison with USART.

Table 351. USART / LPUART features

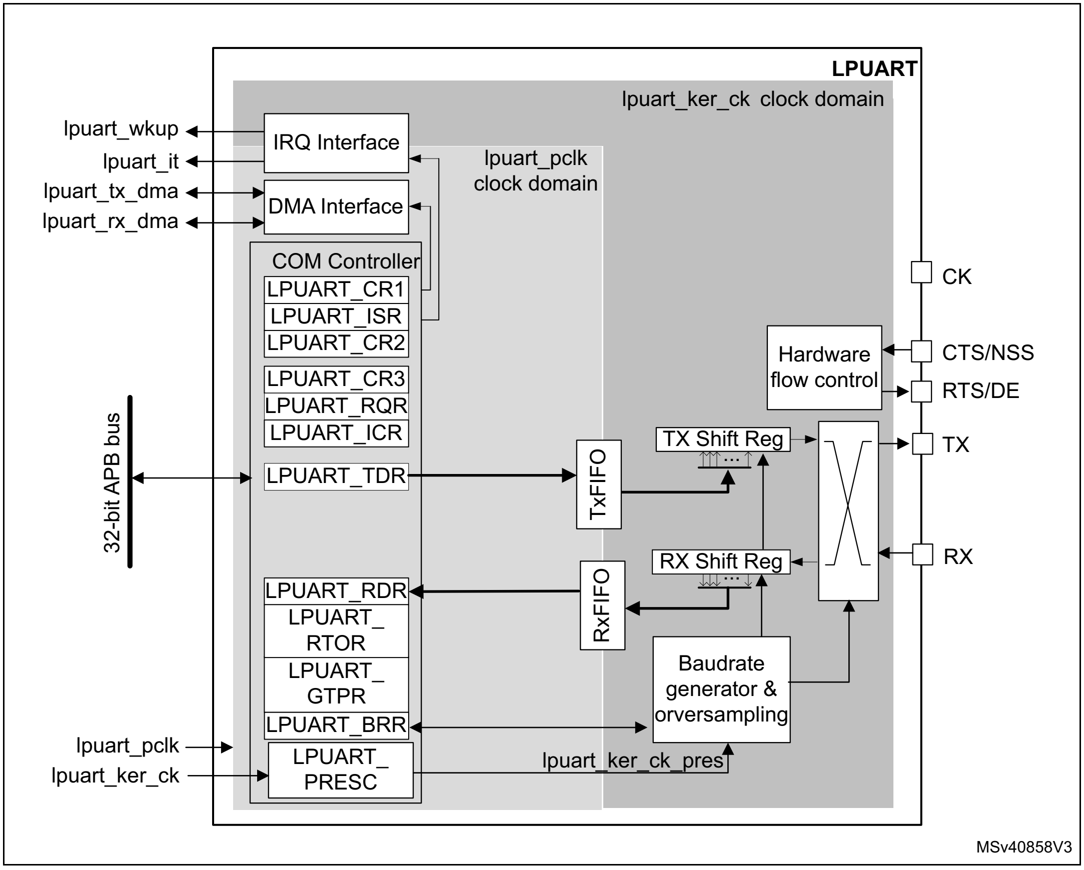
USART / LPUART modes/features[1]	USART1/2/3	UART4/5	LPUART1
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	-	-
Smartcard mode	X	-	-
Single-wire Half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain and wakeup from low-power mode	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver Enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size	8		

1. X = supported.

38.4 LPUART functional description

38.4.1 LPUART block diagram

Figure 557. LPUART block diagram



The simplified block diagram given in *Figure 557* 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.

38.4.2 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.

38.4.3 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 531*).

- 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.

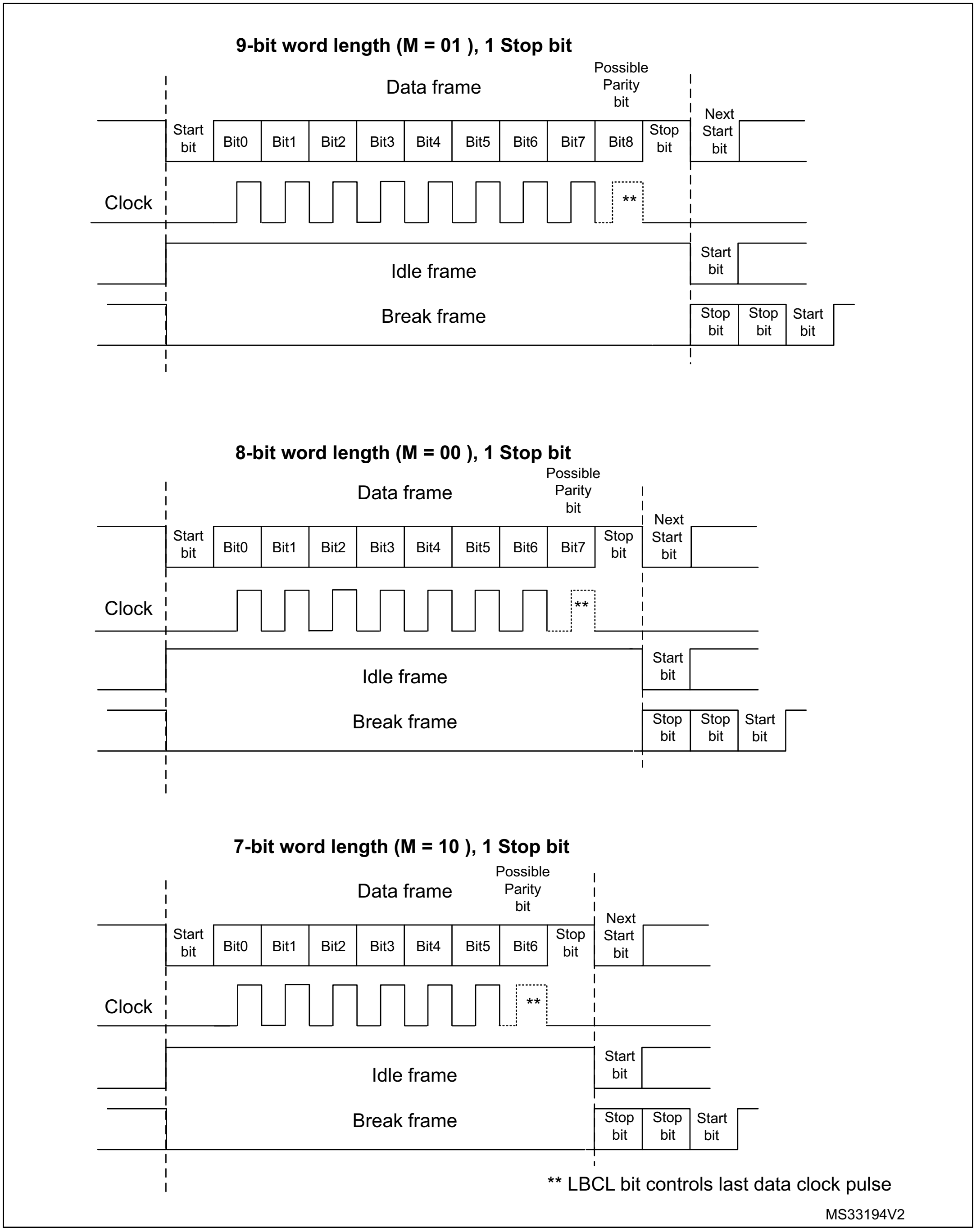
An ***Idle character*** is interpreted as an entire frame of “1”s. (The number of “1” ‘s includes the number of stop bits).

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 558. LPUART word length programming



38.4.4 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.

38.4.5 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 557).

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

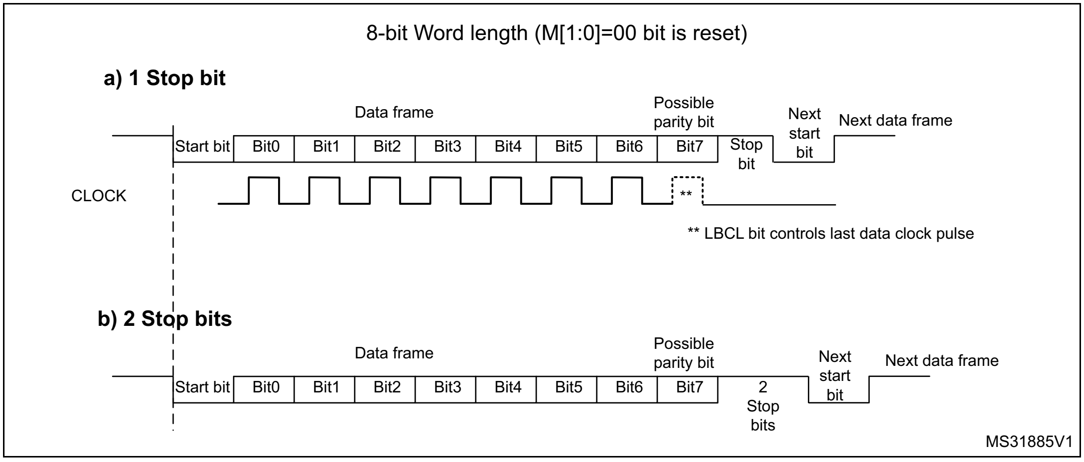
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13,12).

- ***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 559. 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 37.5.10: 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.

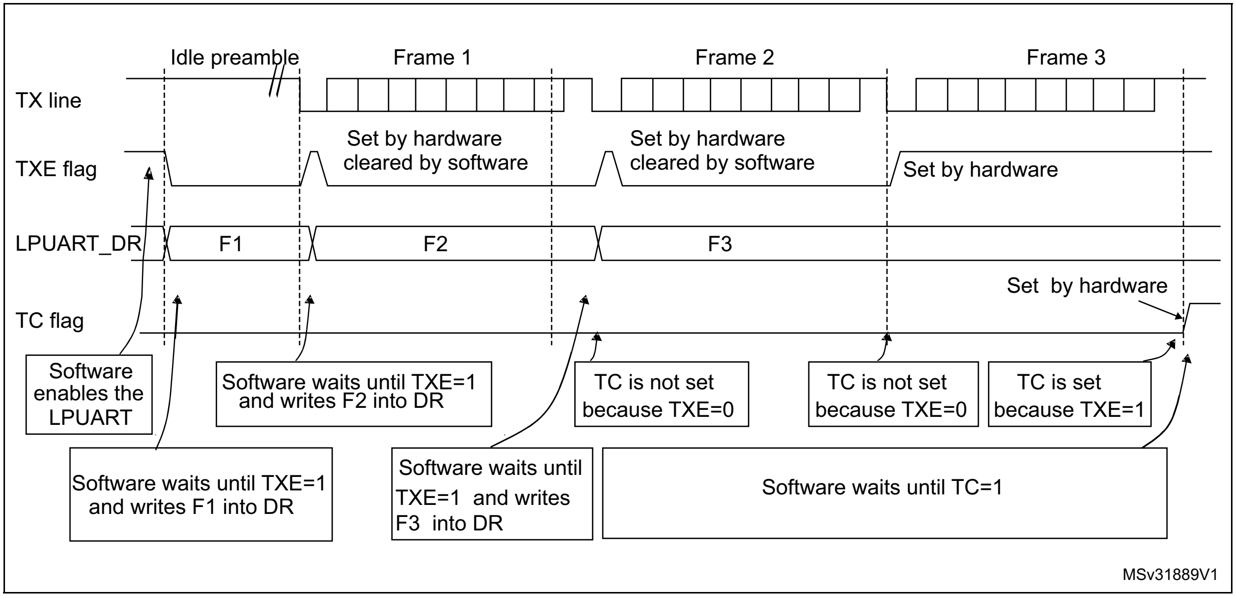
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

  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 560: TC/TXE behavior when transmitting*).

Figure 560. TC/TXE behavior when transmitting


Idle preamble
Frame 1
Frame 2
Frame 3
TX line
TXE flag
Set by hardware
cleared by software
Set by hardware
cleared by software
Set by hardware
LPUART_DR
F1
F2
F3
TC flag
Set by hardware
Software enables the LPUART
Software waits until TXE=1 and writes F2 into DR
TC is not set because TXE=0
TC is not set because TXE=0
TC is set because TXE=1
Software waits until TXE=1 and writes F1 into DR
Software waits until TXE=1 and writes F3 into DR
Software waits until TC=1
MSv31889V1

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 558*).

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.

38.4.6 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 37.5.10: 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 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 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:

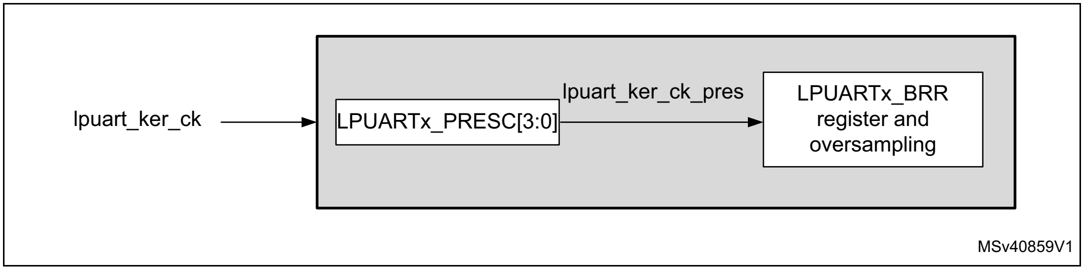
- Possible use of the LPUART in low-power mode
- Communication speed.

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 561. 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 baud as close as possible to the middle of the baud-period. Only a single sample is taken of each of the incoming bauds.

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.

38.4.7 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 352. 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	0.3 KBps	0.3 KBps	0x6D3A	0
2	0.6 KBps	0.6 KBps	0x369D	0
3	1200 Bps	1200.087 Bps	0x1B4E	0.007
4	2400 Bps	2400.17 Bps	0xDA7	0.007
5	4800 Bps	4801.72 Bps	0x6D3	0.035
6	9600 KBps	9608.94 Bps	0x369	0.093

Table 353. 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

38.4.8 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)

$$\mathrm{DTRA} + \mathrm{DQUANT} + \mathrm{DREC} + \mathrm{DTCL} + \mathrm{DWU} < \text{LPUART receiver tolerance}$$

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 354*:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 354. 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 354 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').*

38.4.9 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:

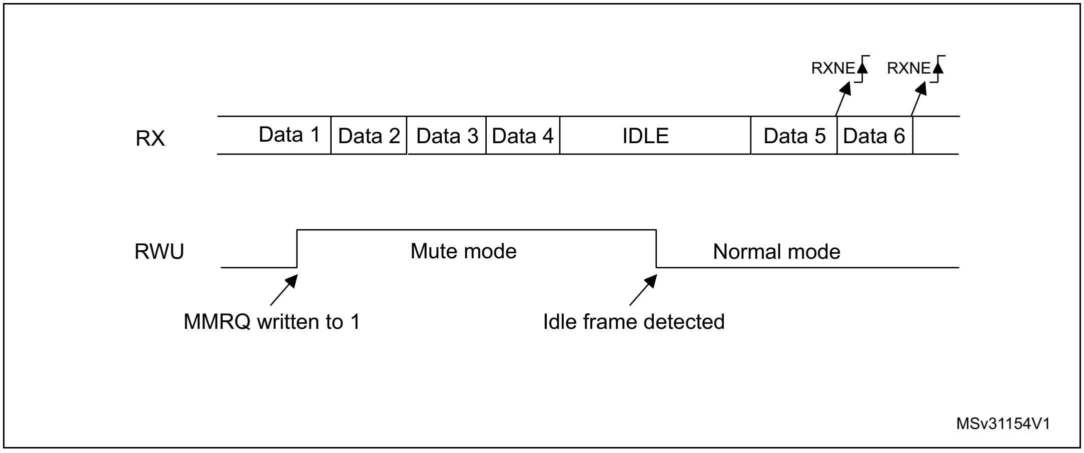
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

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 562*.

Figure 562. 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.

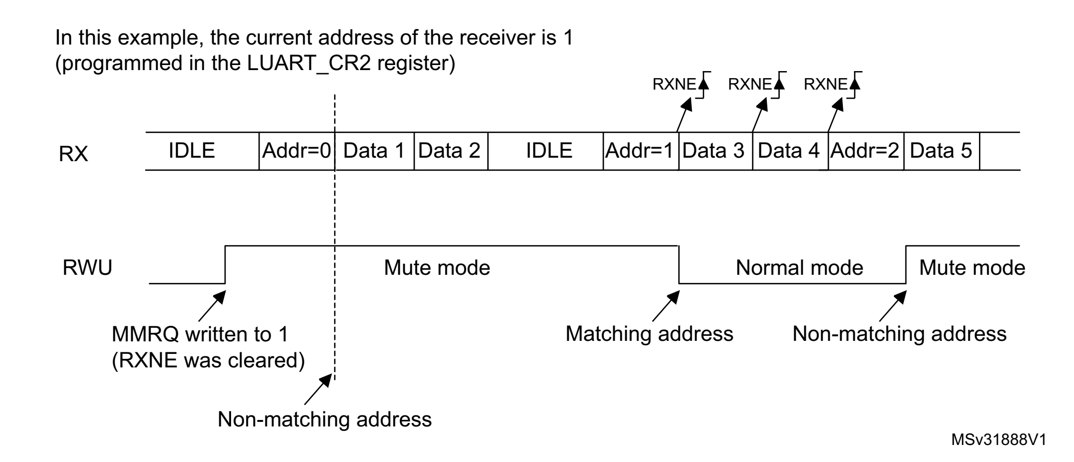
The LPUART also enters Mute mode when the MMRQ bit is written to ‘1’. The RWU bit is also automatically set in this case.

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 563*.

Figure 563. Mute mode using address mark detection



38.4.10 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 355*.

Table 355: 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)).

38.4.11 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.

38.4.12 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 37.4: USART implementation on page 1597 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 37.5.6. To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

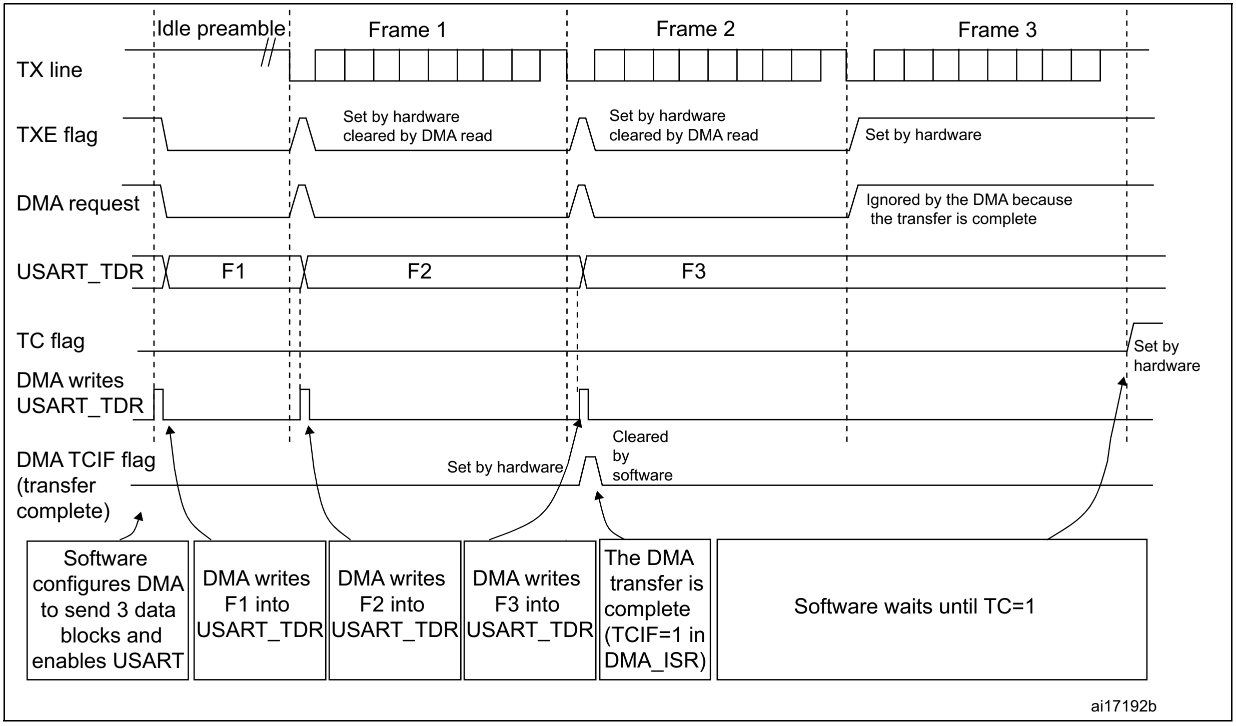
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode 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 564. Transmission using DMA


Idle preamble
Frame 1
Frame 2
Frame 3
TX line
TXE flag
Set by hardware cleared by DMA read
Set by hardware cleared by DMA read
Set by hardware
DMA request
Ignored by the DMA because the transfer is complete
USART_TDR
F1
F2
F3
TC flag
Set by hardware
DMA writes USART_TDR
DMA TCIF flag (transfer complete)
Set by hardware
Cleared by software
Software configures DMA to send 3 data blocks and enables USART
DMA writes F1 into USART_TDR
DMA writes F2 into USART_TDR
DMA writes F3 into USART_TDR
The DMA transfer is complete (TCIF=1 in DMA_ISR)
Software waits until TC=1
ai17192b

Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

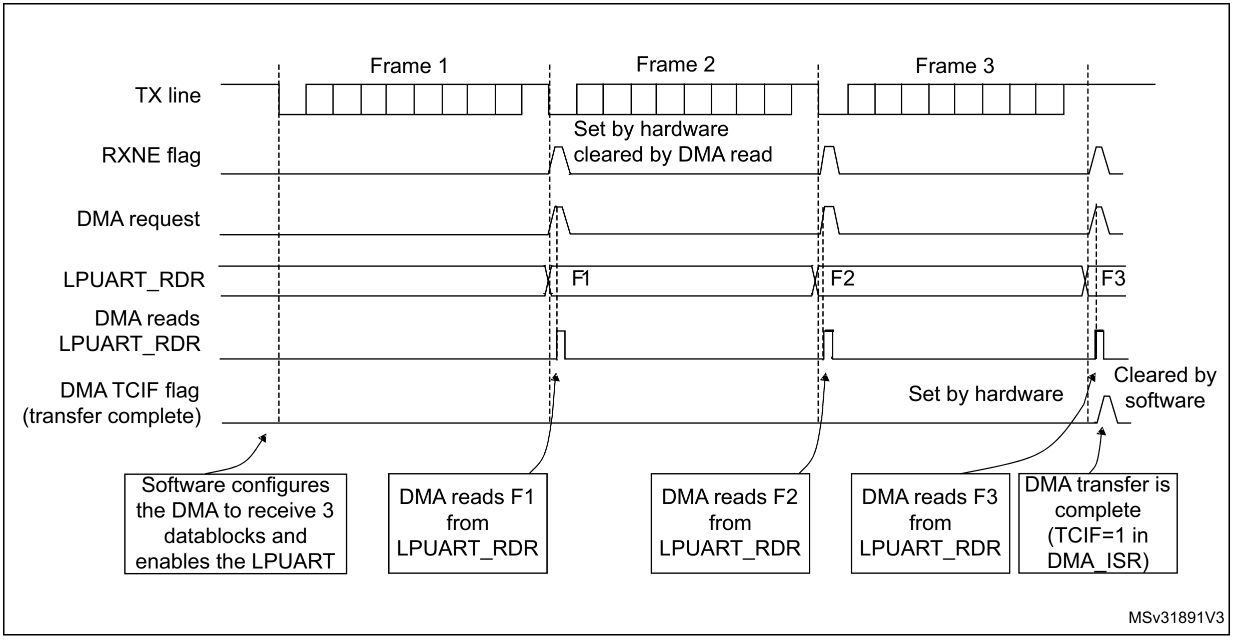
Reception using DMA

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 565. Reception using DMA


Frame 1
Frame 2
Frame 3
TX line
RXNE flag
Set by hardware
cleared by DMA read
DMA request
LPUART_RDR
F1
F2
F3
DMA reads
LPUART_RDR
DMA TCIF flag
(transfer complete)
Set by hardware
Cleared by software
Software configures the DMA to receive 3 datablocks and enables the LPUART
DMA reads F1 from LPUART_RDR
DMA reads F2 from LPUART_RDR
DMA reads F3 from LPUART_RDR
DMA transfer is complete (TCIF=1 in DMA_ISR)
MSv31891V3

Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

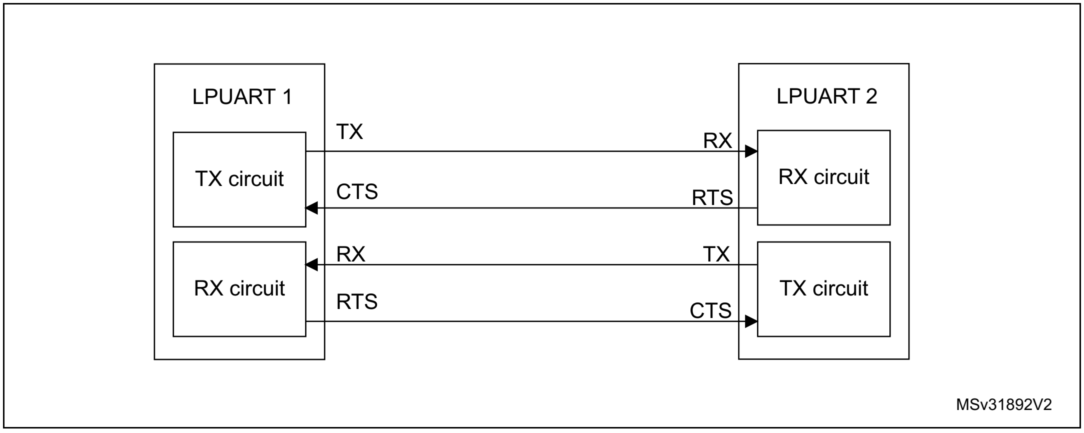
Error flagging and interrupt generation in multibuffer communication

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.

38.4.13 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The *Figure 552* shows how to connect 2 devices in this mode:

Figure 566. Hardware flow control between 2 LPUARTs

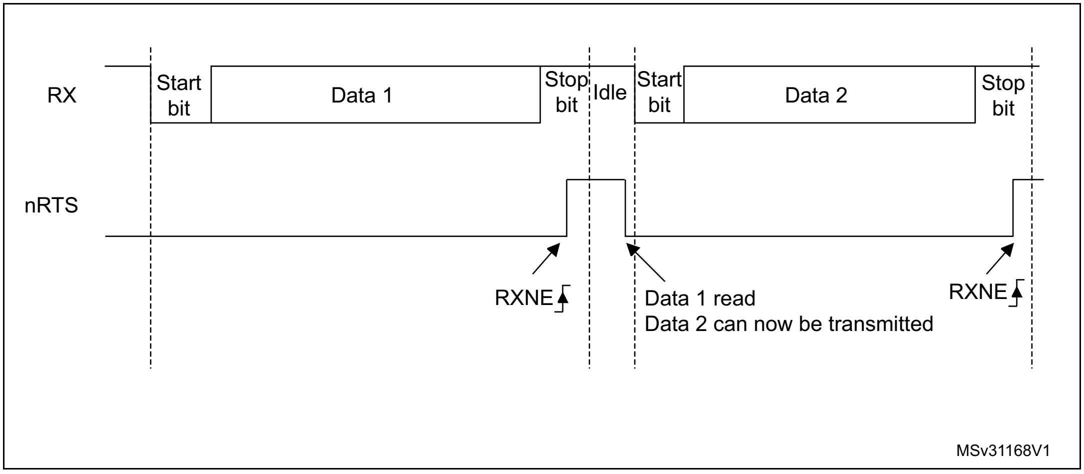


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then nRTS is asserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 567* shows an example of communication with RTS flow control enabled.

Figure 567. RS232 RTS flow control



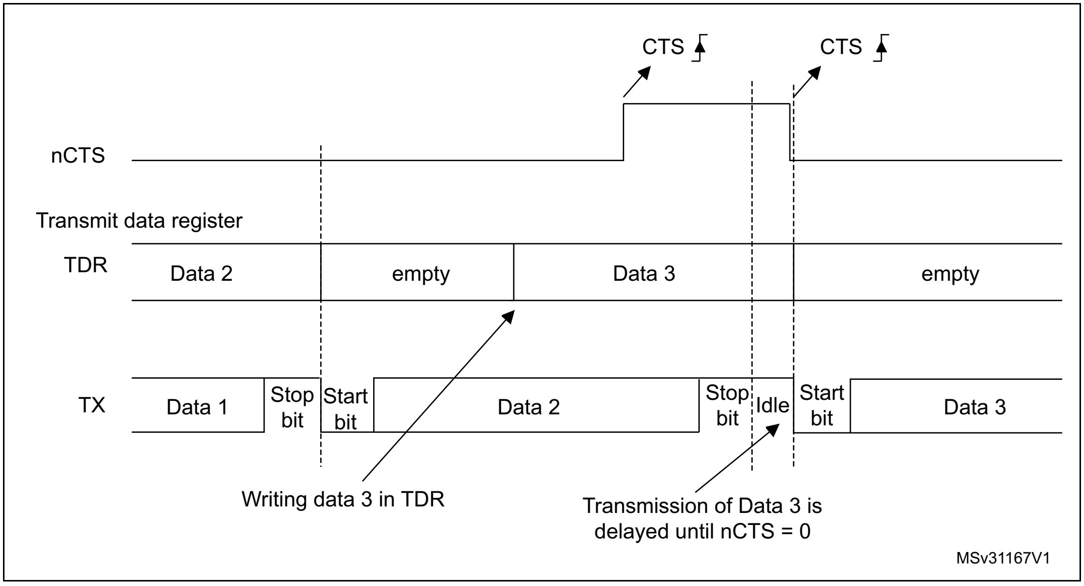
Note: When FIFO mode is enabled, nRTS is deasserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (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 nCTS is deasserted 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 nCTS 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 568* shows an example of communication with CTS flow control enabled.

Figure 568. RS232 CTS flow control



Note: For correct behavior, nCTS must be asserted 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]

38.4.14 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 usart_pclk is gated, the LPUART 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

  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. In this case the **lpuart_wkup** interrupt is not mandatory for the wakeup. The WUF being set is sufficient to wakeup the MCU from low-power mode.

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 usart_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 569 shows the behavior when the wakeup event is verified.

Figure 569. Wakeup event verified (wakeup event = address match, FIFO disabled)

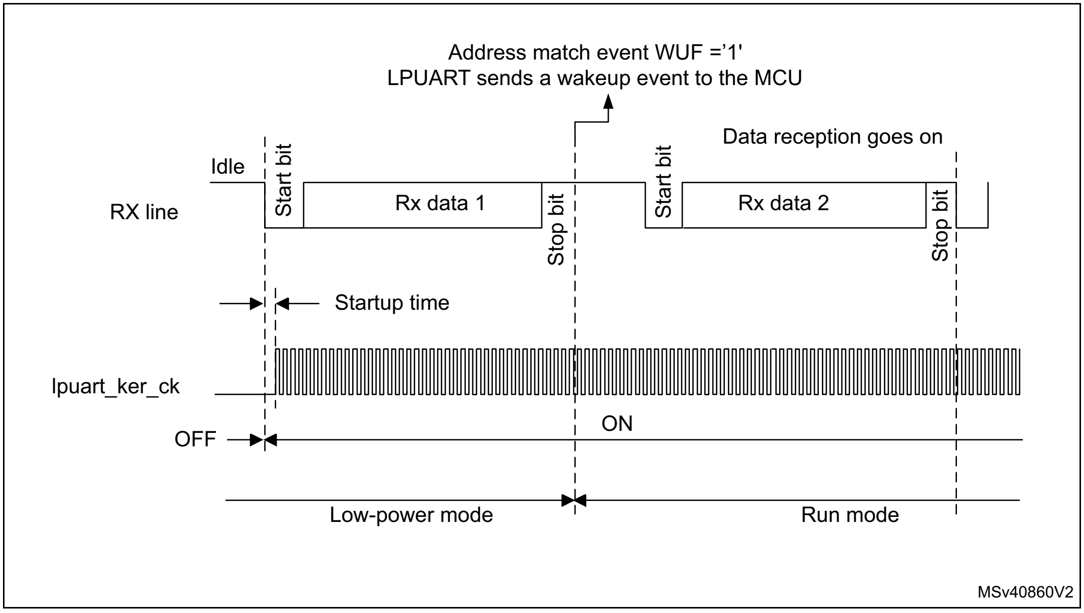
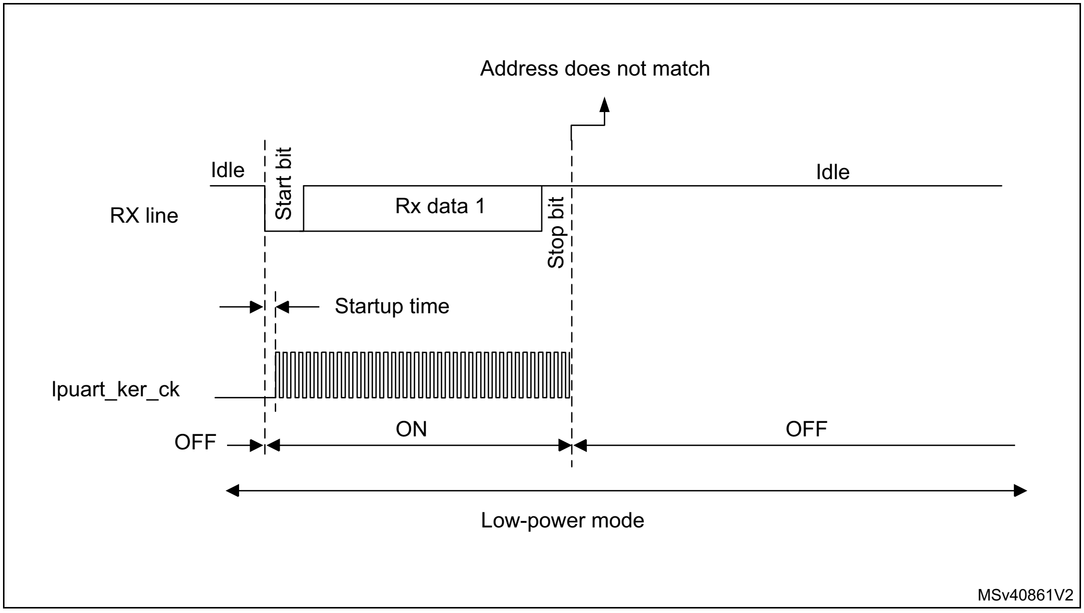


Figure 570 shows the behavior when the wakeup event is not verified.

Figure 570. 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 38.4.8: 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 354: 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 HSI16 is used as lpuart_ker_ck, and the HSI16 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.

38.5 LPUART in low-power modes

Table 356. 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 38.3: 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.

38.6 LPUART interrupts

Refer to *Table 357* for a detailed description of all LPUART interrupt requests.

Table 357. 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 38.3: 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.

38.7 LPUART registers

Refer to *Section 1.2 on page 73* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

38.7.1 LPUART control register 1 [alternate] (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 37.5.20: 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 38.4.13: 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 HSI16 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 HSI16 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.

38.7.2 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 sec**tion).**

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	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 37.5.20: 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 38.4.13: 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 HSI16 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 HSI16 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.

38.7.3 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

ADD[7:4]:

These bits give the address of the LPUART node or a character code to be recognized.

They are used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or Stop mode. The MSB of the character sent by the transmitter should be equal to 1. They can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.

These bits can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE = 0)

ADD[3:0]:

These bits give the address of the LPUART node or a character code to be recognized.

They are used for wakeup with address mark detection in multiprocessor communication during Mute mode or low-power mode.

These bits can only be written when reception is disabled (RE = 0) or the LPUART 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.

38.7.4 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 37.4: 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 37.4: 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 nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.
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 nRTS output is asserted (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.

38.7.5 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].

38.7.6 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.

38.7.7 LPUART interrupt and status register [alternate] (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 nCTS input pin.

0: nCTS line set

1: nCTS 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 nCTS 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 nCTS status line

1: A change occurred on the nCTS 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.

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_CR1 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.

38.7.8 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 nCTS input pin.

0: nCTS line set

1: nCTS 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 nCTS 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 nCTS status line

1: A change occurred on the nCTS 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.

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_CR1 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

38.7.9 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 37.4: 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.

38.7.10 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 557*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

38.7.11 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 557*).

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.

38.7.12 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.

38.7.13 LPUART register map

The table below gives the LPUART register map and reset values.

Table 358. 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 358. 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 2.2: Memory organization* for the register boundary addresses.

39 Serial peripheral interface / integrated interchip sound (SPI/I2S)

39.1 Introduction

The SPI/I²S interface can be used to communicate with external devices using the SPI protocol or the I^{2}S audio protocol. SPI or I^{2}S 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^{2}S) 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^{2}S standard, the MSB- and LSB-justified standards and the PCM standard.

39.2 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

39.3 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. Ratio is fixed at 256 × F_S (where F_S is the audio sampling frequency)

39.4 SPI/I2S implementation

The following table describes all the SPI instances and their features embedded in the devices.

Table 359. STM32G4 Series SPI and SPI/I2S implementation

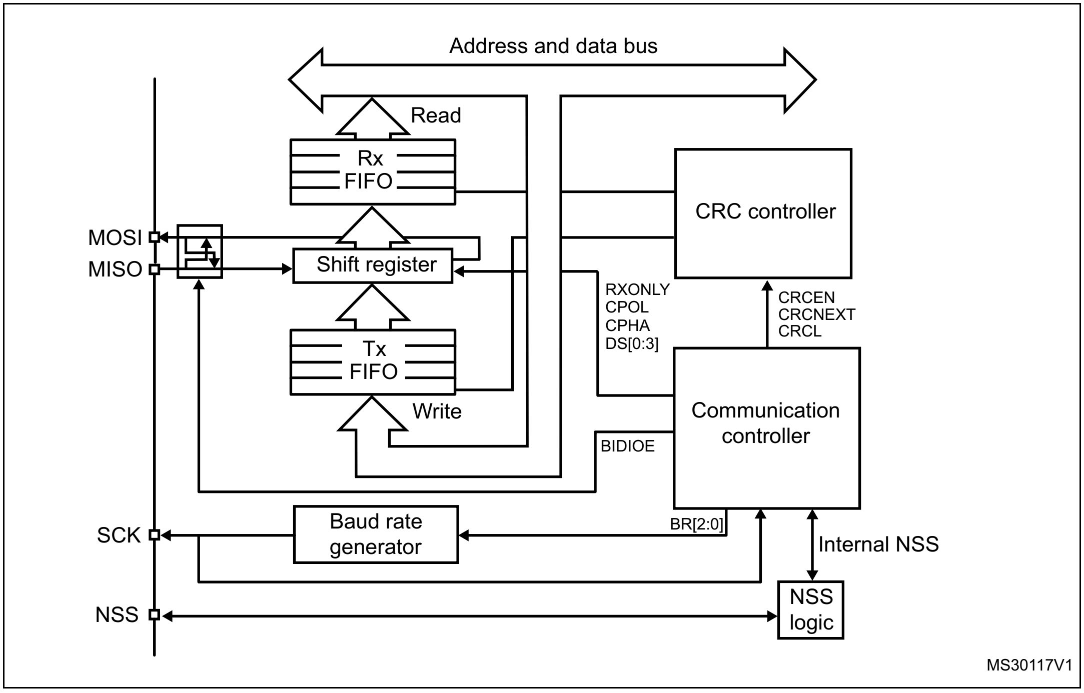
SPI features	SPI1	SPI2S2	SPI2S3	SPI4
Enhanced NSSP & TI modes	Yes	Yes	Yes	Yes
Hardware CRC calculation	Yes	Yes	Yes	Yes
I2S support	No	Yes	Yes	No
Data size configurable	from 4 to 16-bit	from 4 to 16-bit	from 4 to 16-bit	from 4 to 16-bit
Rx/Tx FIFO size	32-bit	32-bit	32-bit	32-bit
Wakeup capability from Low-power Sleep	Yes	Yes	Yes	Yes

39.5 SPI functional description

39.5.1 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 571*.

Figure 571. 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 39.5.5: 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.

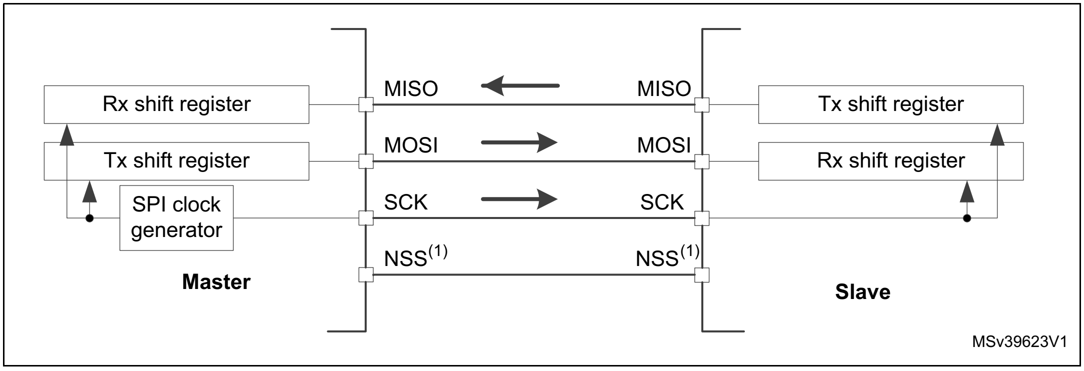
39.5.2 Communications between one master and one slave

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 572. Full-duplex single master/ single slave application

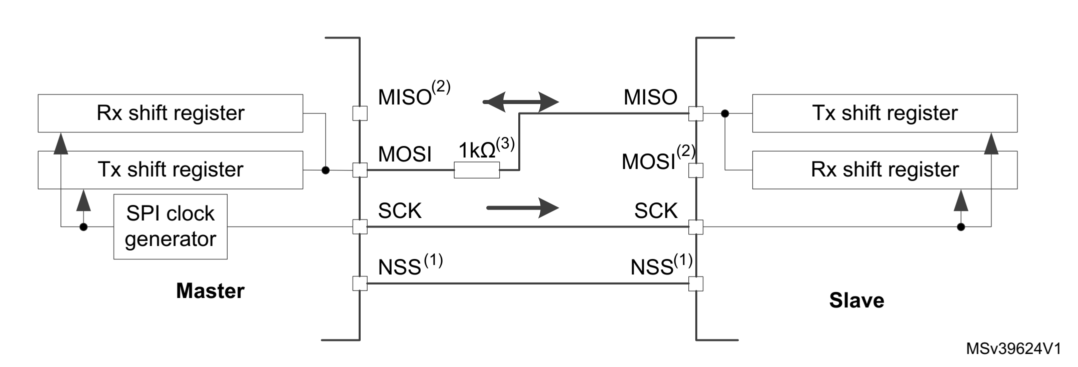


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 39.5.5: Slave select (NSS) pin management*.

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 573. Half-duplex single master/ single slave application



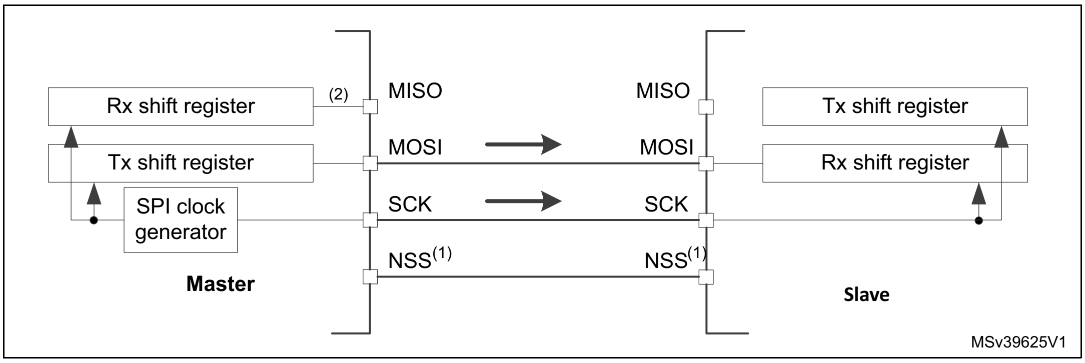
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 39.5.5: Slave select (NSS) pin management*.
2. In this 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 *39.5.5: 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 574. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)



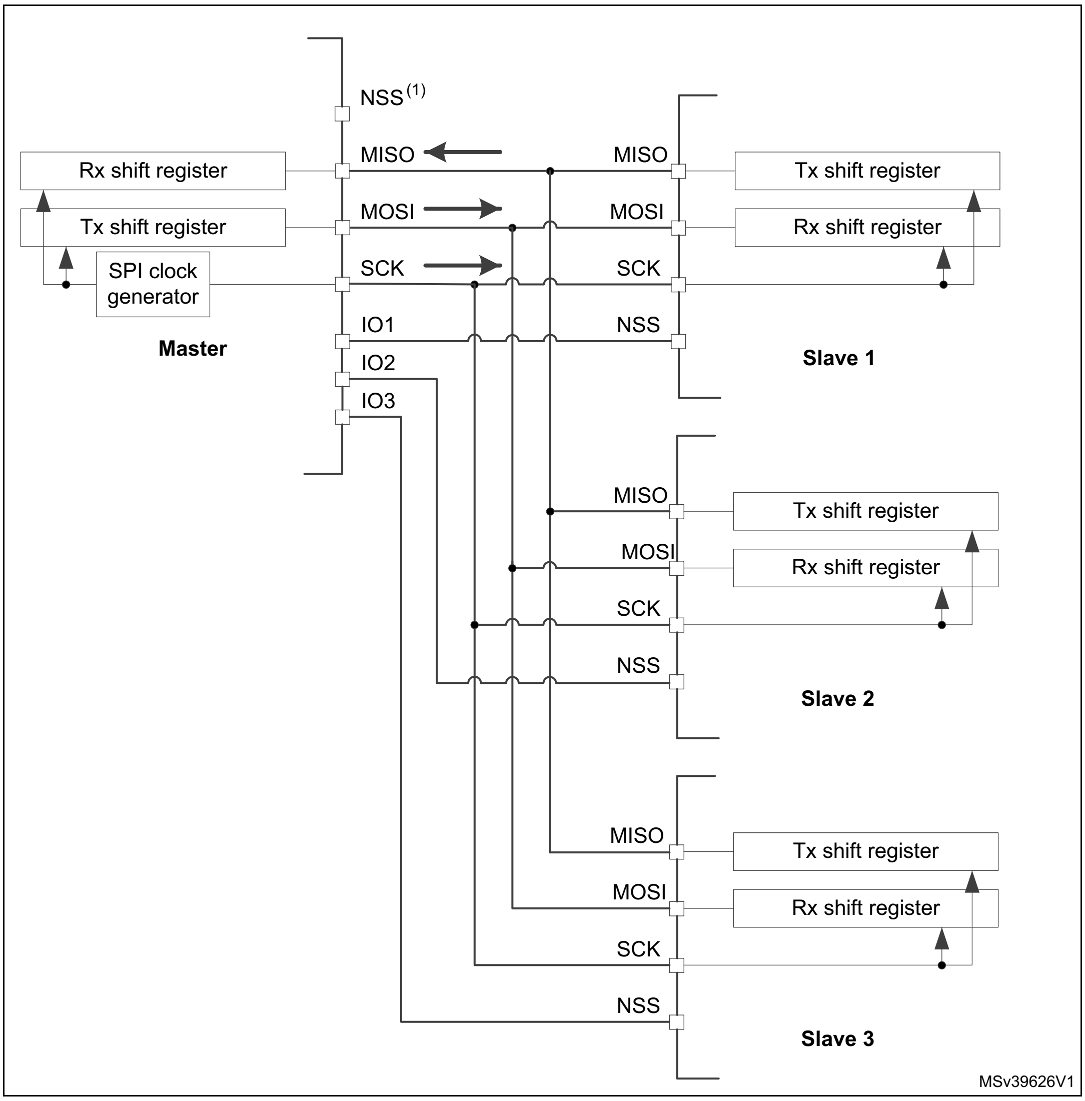
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 39.5.5: Slave select (NSS) pin management*.
2. An accidental 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. OVF 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).

39.5.3 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 575*). 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 575. 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)).

39.5.4 Multi-master communication

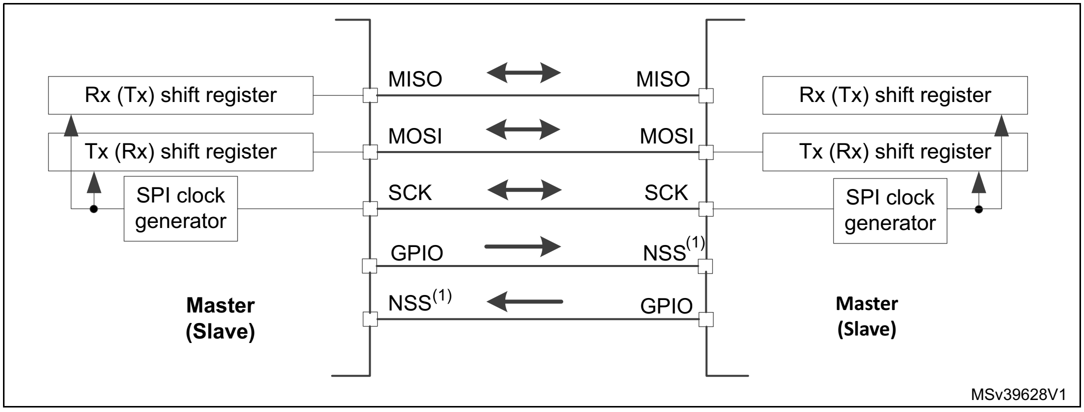
Unless SPI bus is not designed for a multi-master capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

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 576. 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.

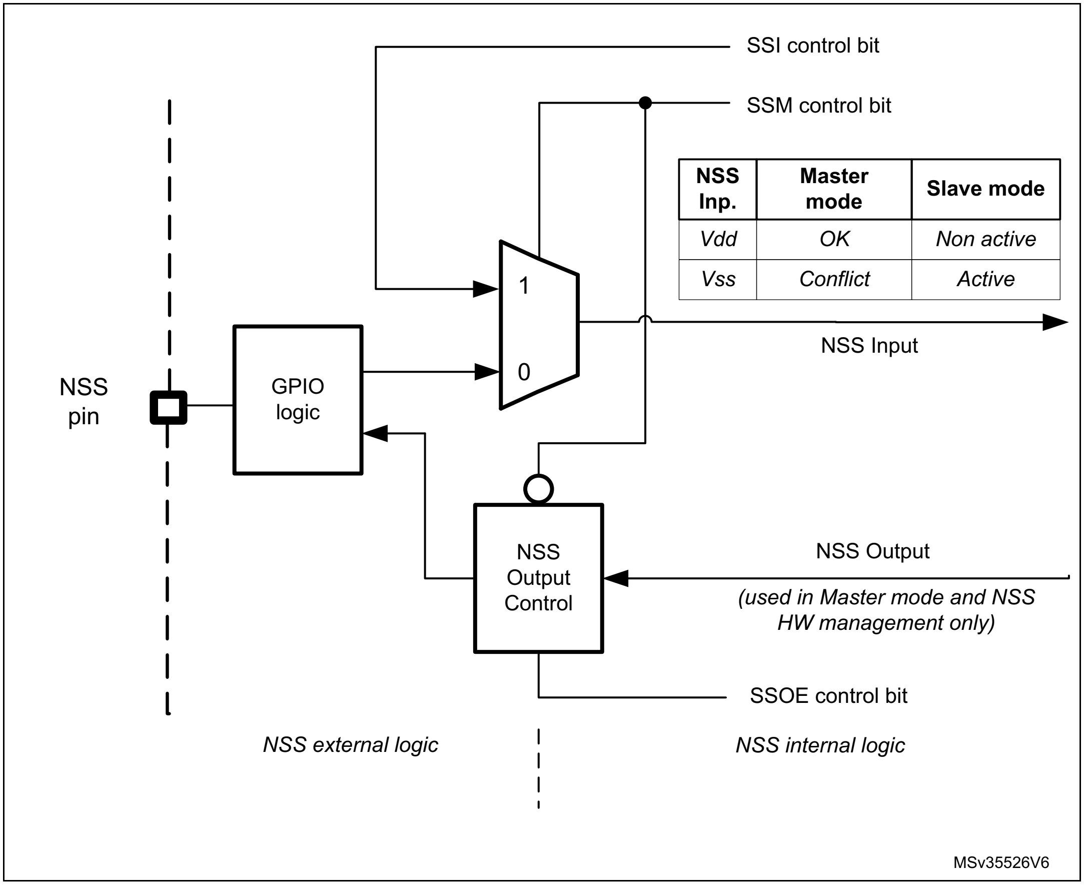
39.5.5 Slave select (NSS) pin management

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 577. Hardware/software slave select management



NSS Inp.	Master mode	Slave mode
Vdd	*OK*	*Non active*
Vss	*Conflict*	*Active*

39.5.6 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.

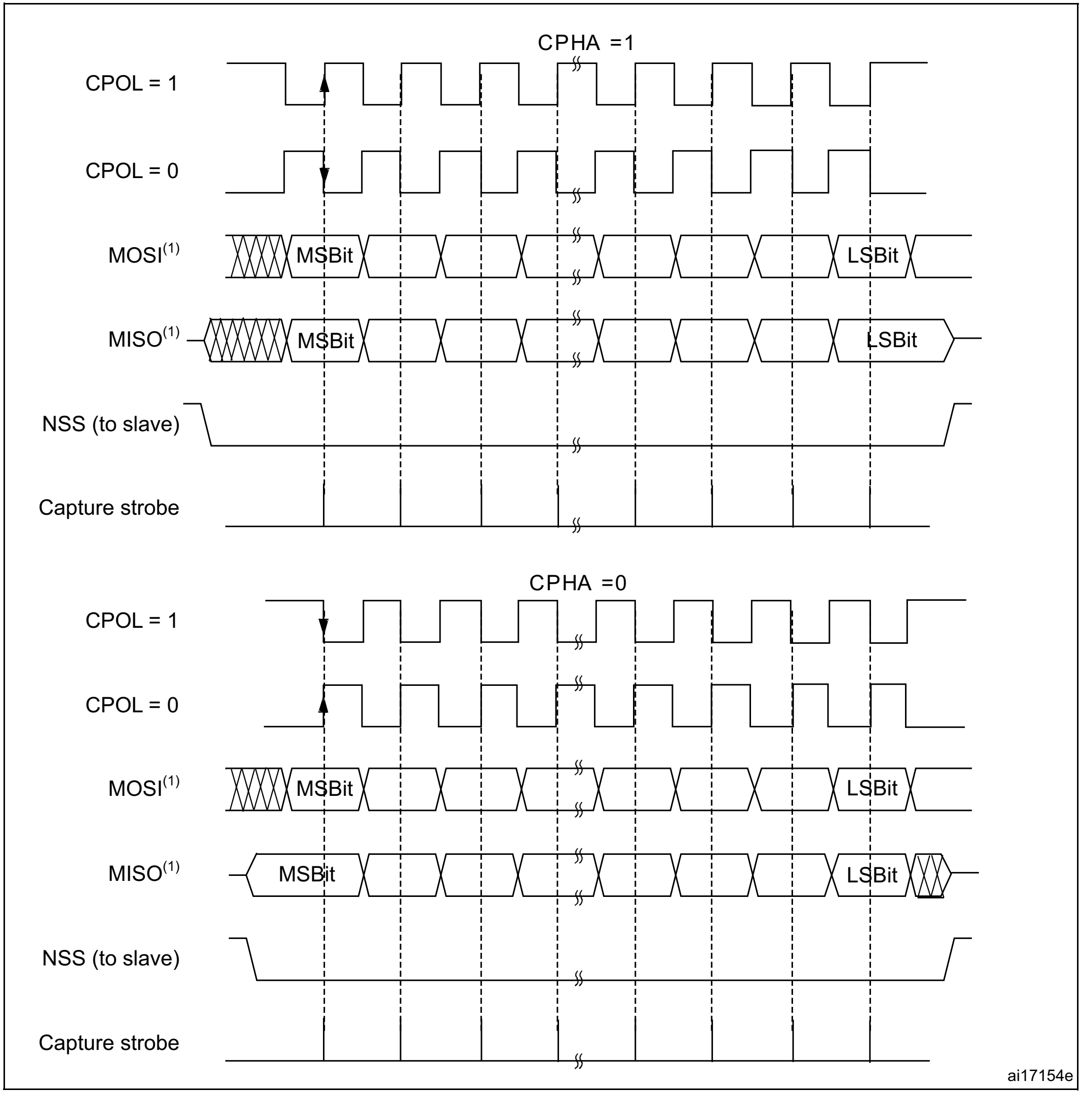
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 578, 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 578. Data clock timing diagram

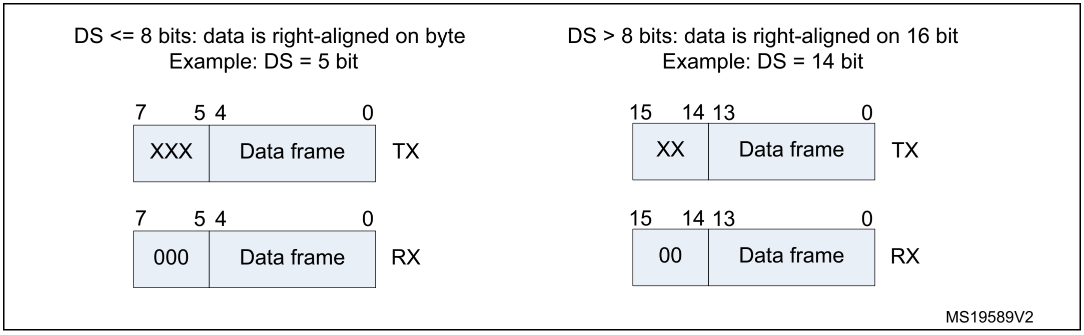


1. The order of data bits depends on LSBFIRST bit setting.

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 579*). During communication, only bits within the data frame are clocked and transferred.

Figure 579. 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.

39.5.7 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

39.5.8 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.

39.5.9 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 39.5.14: 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 39.5.13: 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 581* through *Figure 584*.

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 39.5.10: 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 39.5.5: 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.

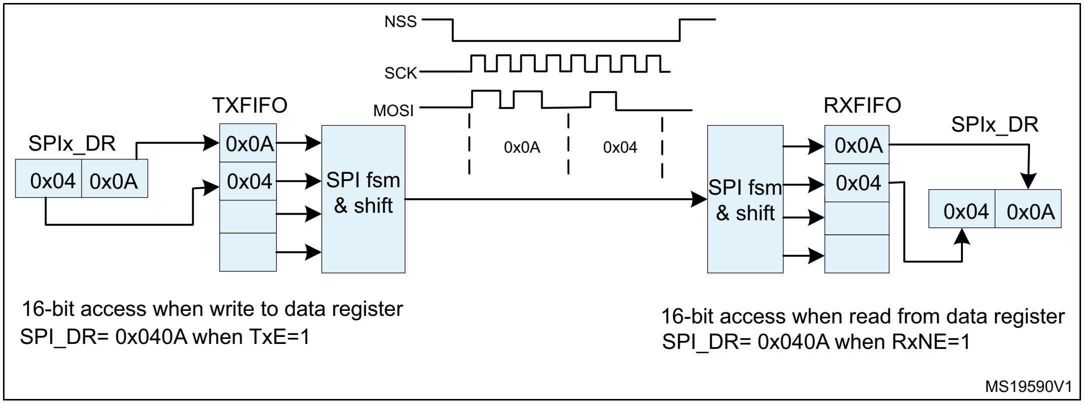
Data packing

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 580* 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 580. Packing data in FIFO for transmission and reception


NSS
SCK
MOSI
0x0A
0x04
TXFIFO
SPIx_DR
0x04
0x0A
0x0A
0x04
SPI fsm & shift
RXFIFO
SPIx_DR
0x0A
0x04
SPI fsm & shift
0x04
0x0A
16-bit access when write to data register
SPI_DR= 0x040A when TxE=1
16-bit access when read from data register
SPI_DR= 0x040A when RxNE=1
MS19590V1

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 TXE or RXNE 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 581* through *Figure 584*.

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 1748*.)

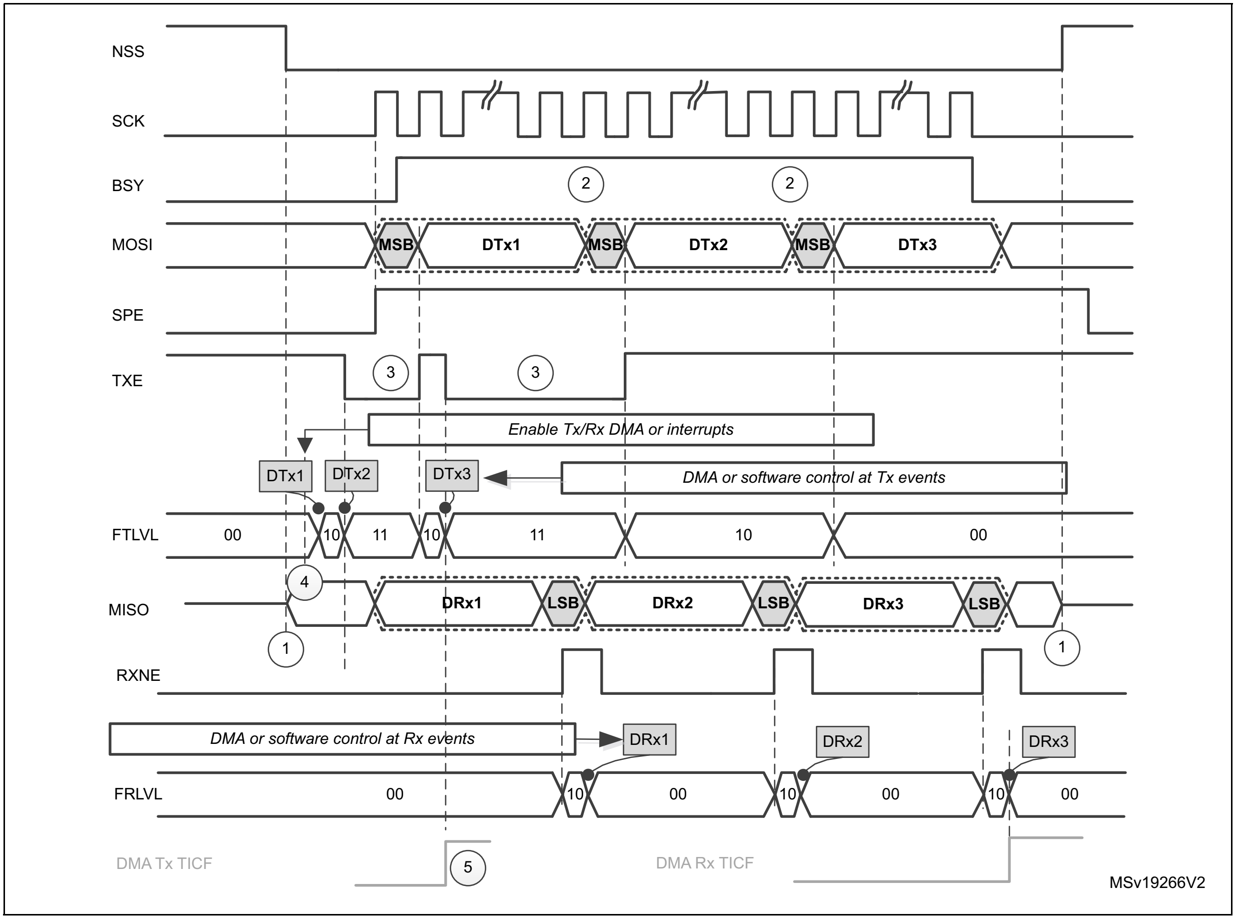
Communication diagrams

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 581 on page 1752* through *Figure 584 on page 1755*:

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 581. Master full-duplex communication



Assumptions for master full-duplex communication example:

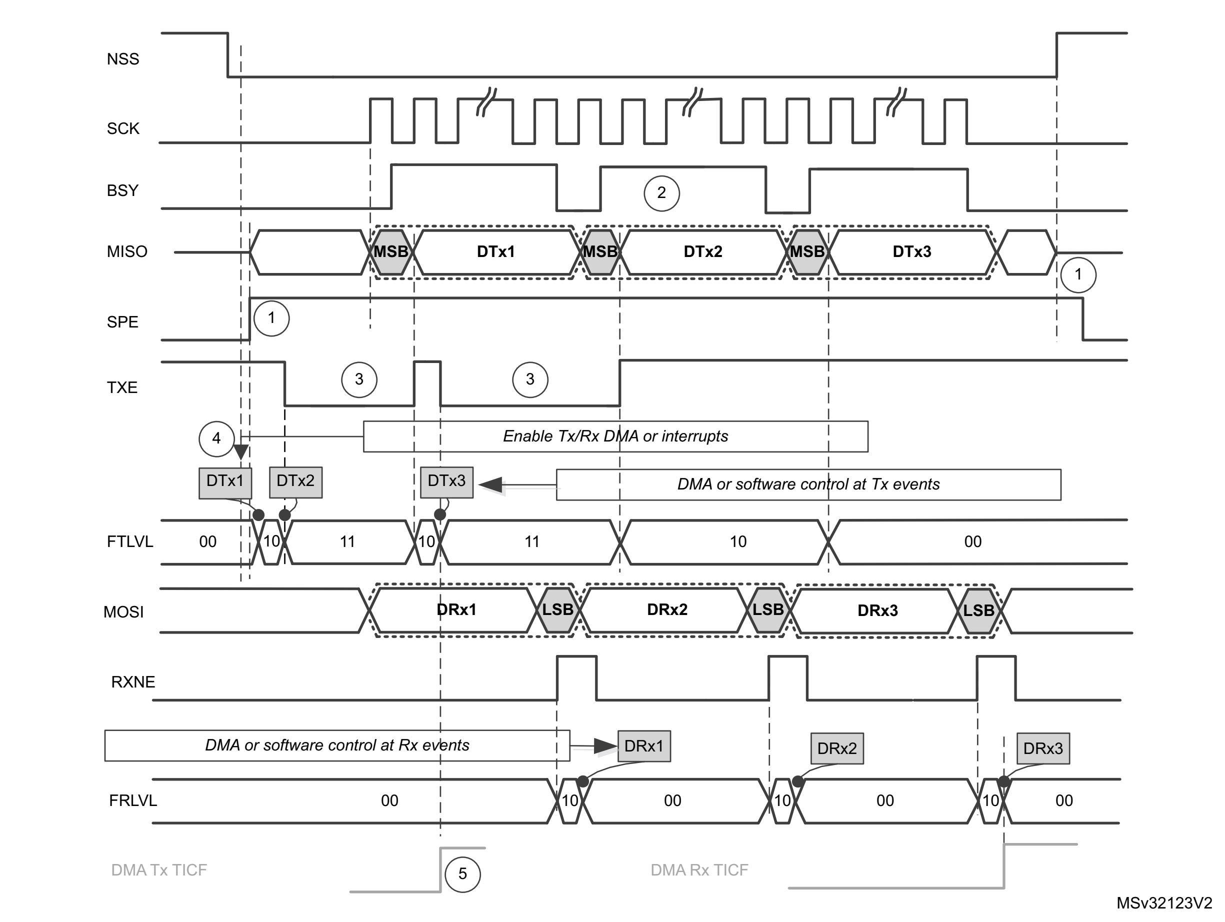
- Data size > 8 bit

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 1751* for details about common assumptions and notes.

Figure 582. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

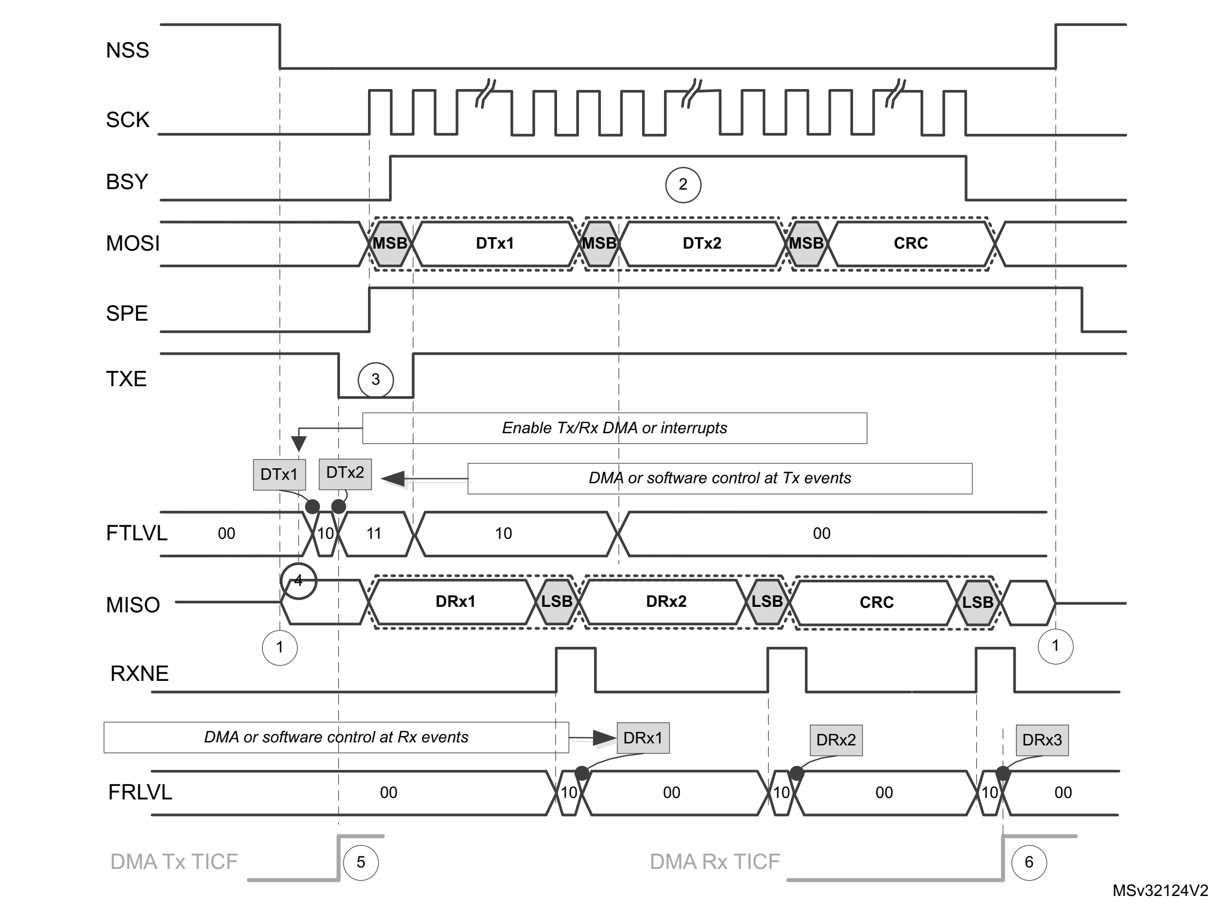
- Data size > 8 bit

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 1751* for details about common assumptions and notes.

Figure 583. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

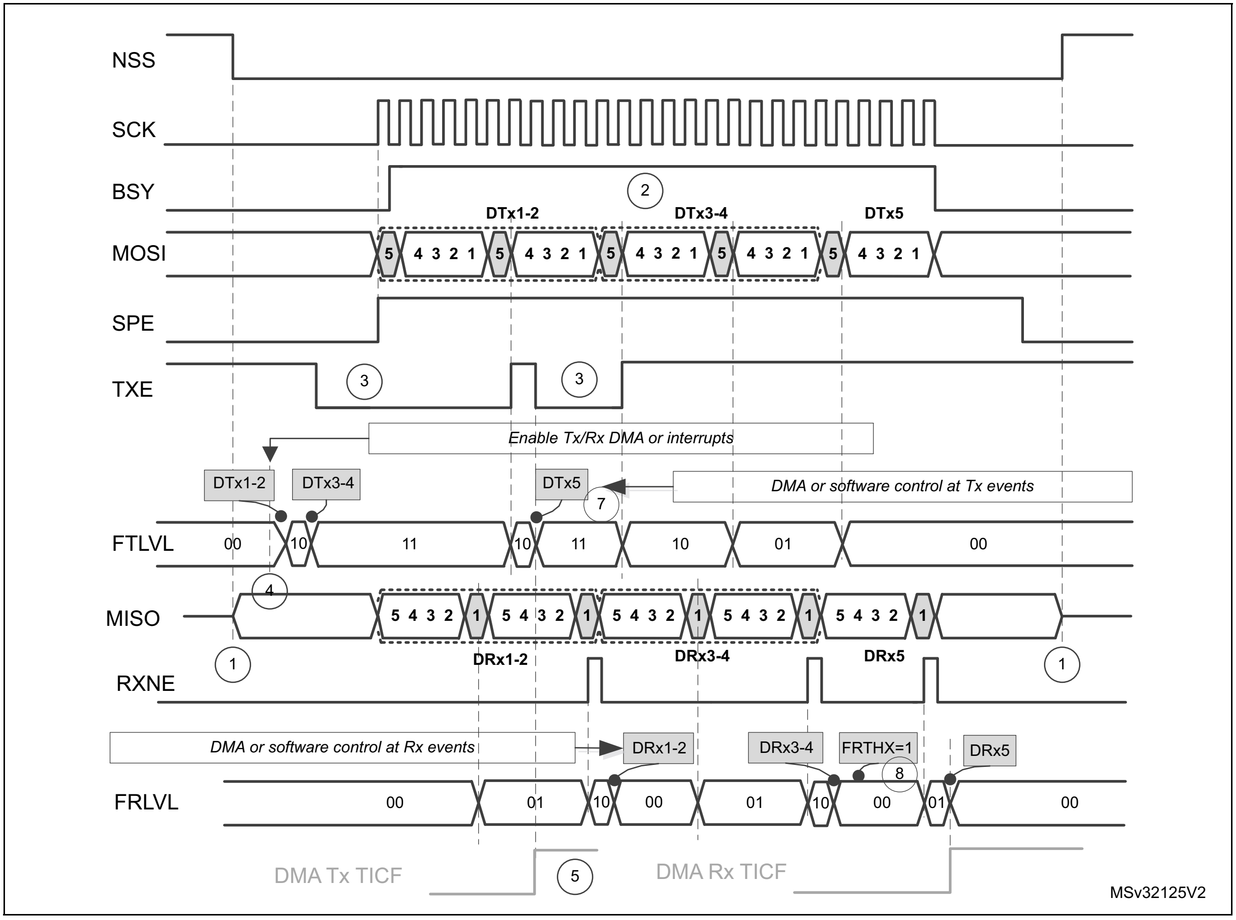
- Data size = 16 bit
- CRC enabled

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 1751* for details about common assumptions and notes.

Figure 584. 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 1751* for details about common assumptions and notes.

39.5.10 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.

39.5.11 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 39.5.14: 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.

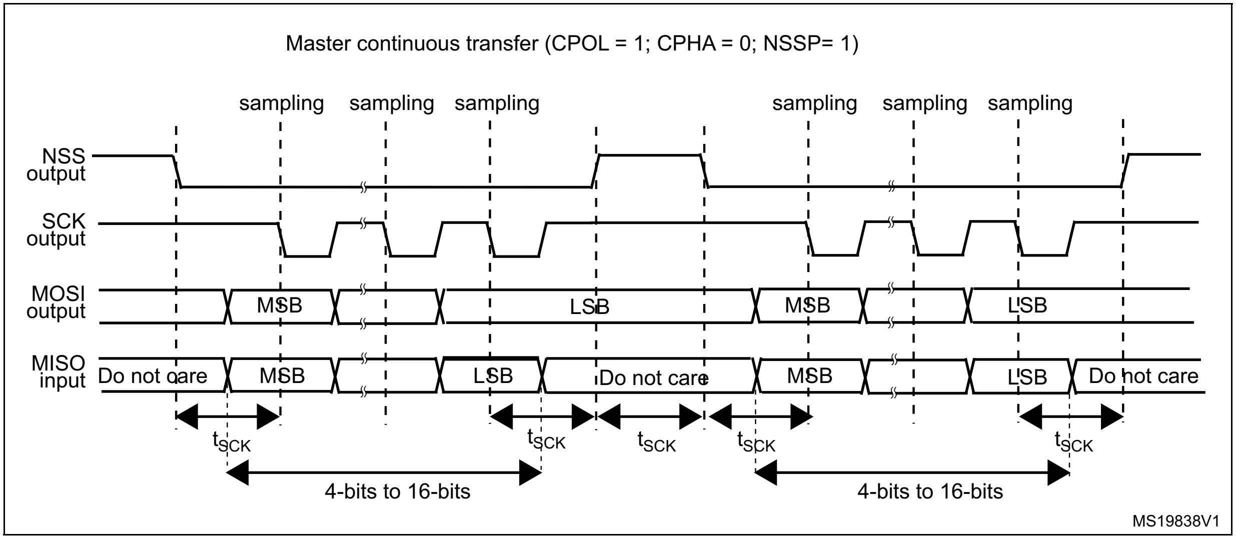
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

39.5.12 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 585 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 585. 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.

39.5.13 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 586*). 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:

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

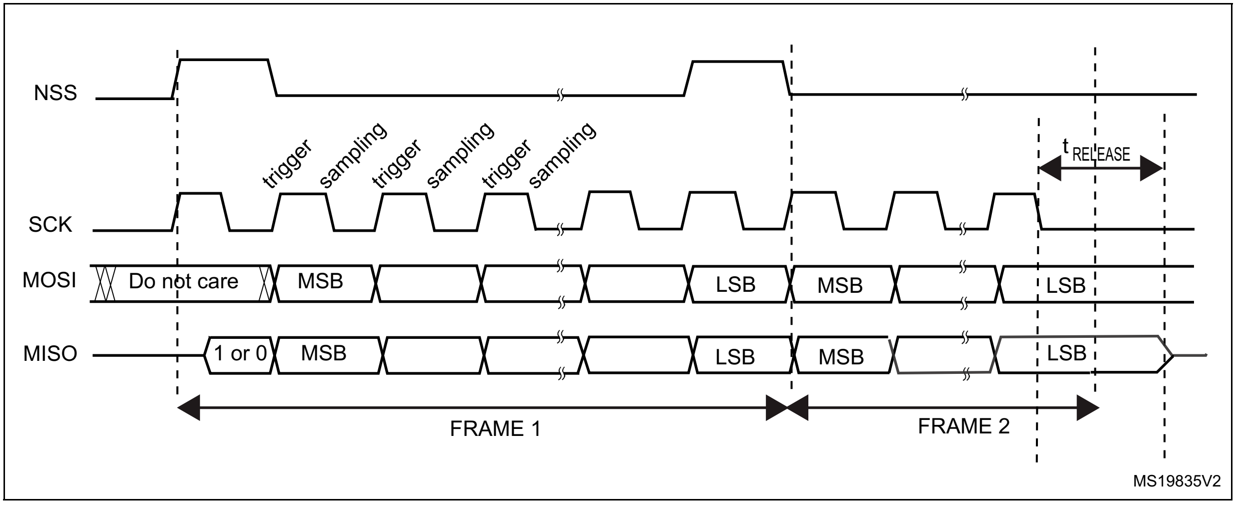
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

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 586: TI mode transfer shows the SPI communication waveforms when TI mode is selected.

Figure 586. TI mode transfer



39.5.14 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.

39.6 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 360. SPI interrupt requests

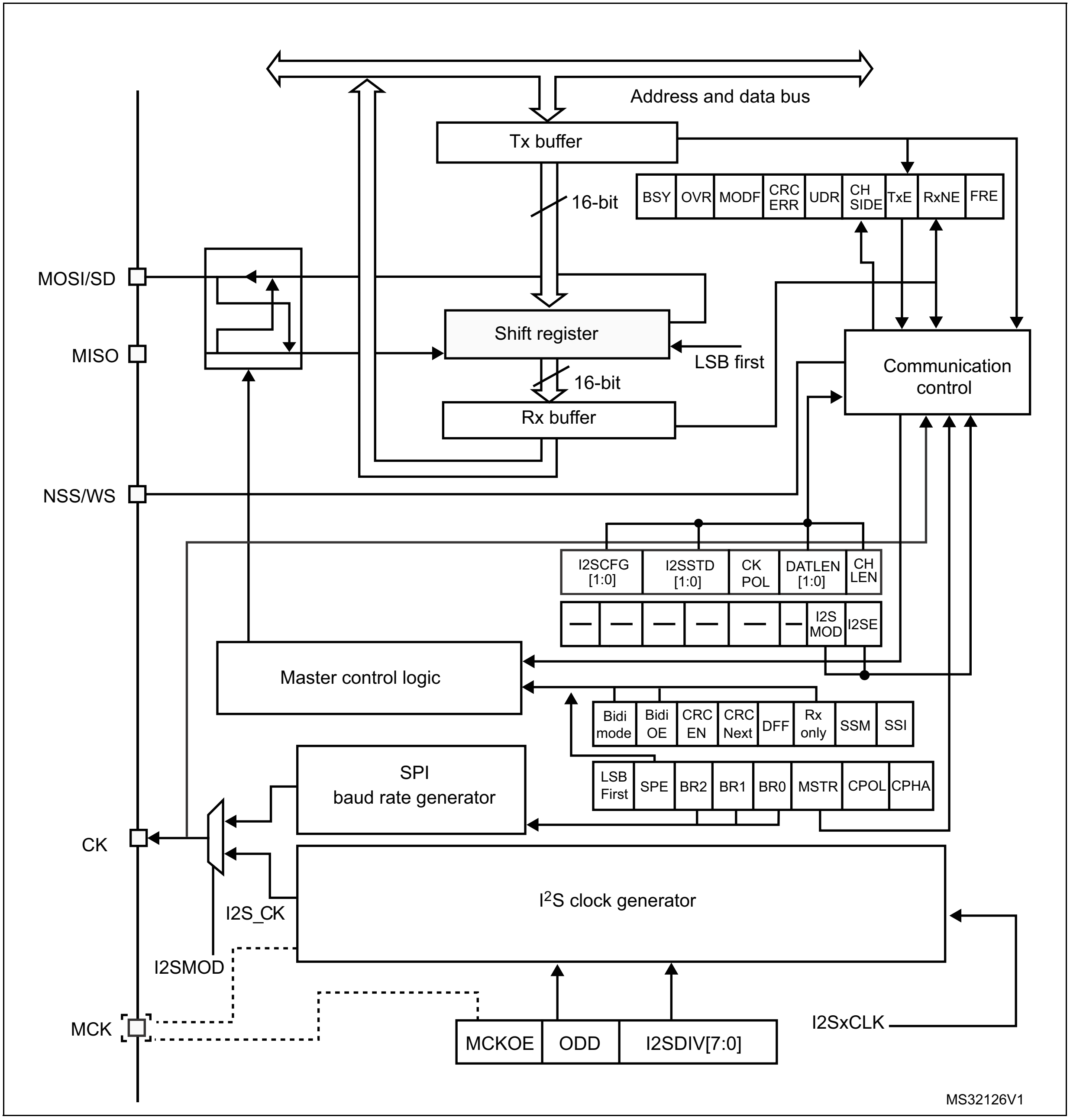
Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	
CRC protocol error	CRCERR	

39.7 I2S functional description

39.7.1 I2S general description

The block diagram of the I2S is shown in *Figure 587*.

Figure 587. 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, 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.

39.7.2 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).

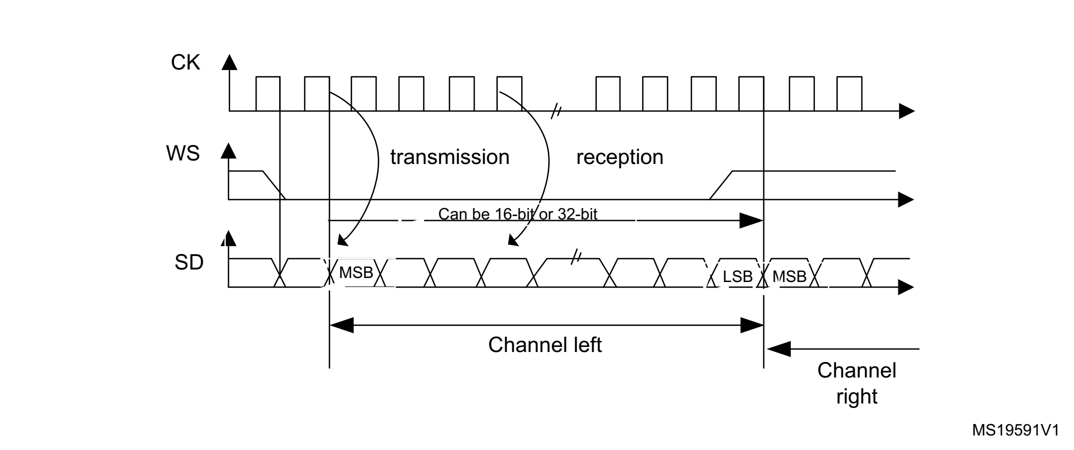
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

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

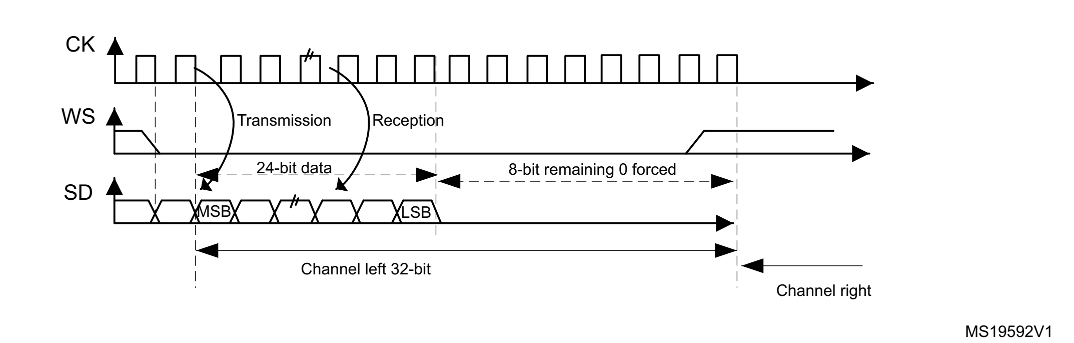
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

Figure 588. 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 589. I^{2}S Philips standard waveforms (24-bit frame)

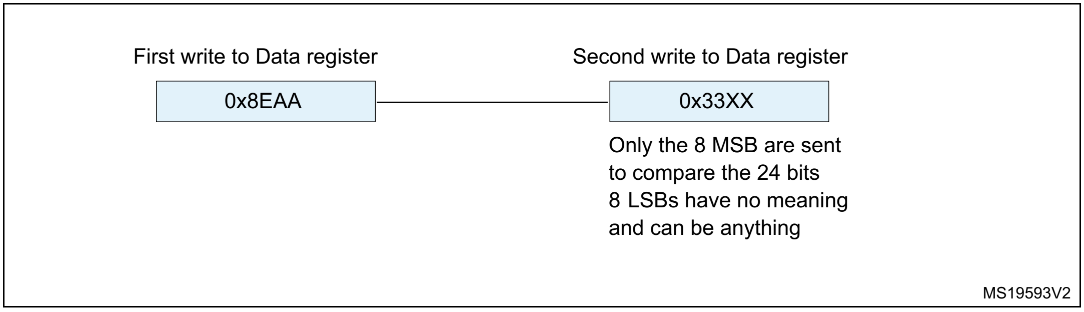


This mode needs two write or read operations to/from the SPIx_DR register.

- In transmission mode:

  If 0x8EAA33 has to be sent (24-bit):

Figure 590. Transmitting 0x8EAA33

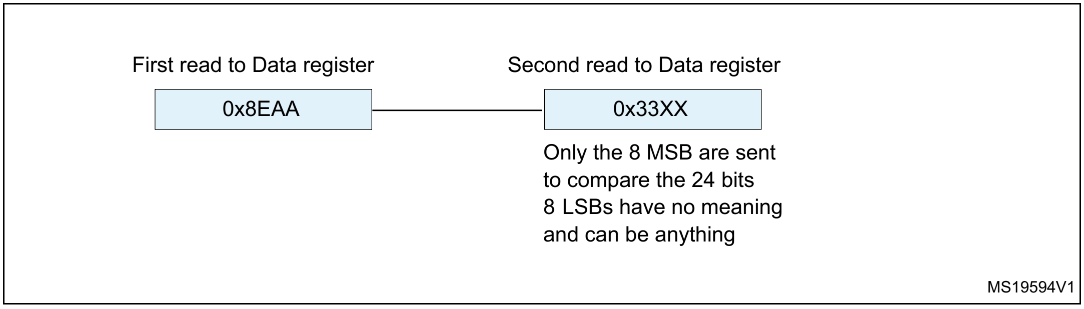


Only the 8 MSB are sent to compare the 24 bits 8 LSBs have no meaning and can be anything

- In reception mode:

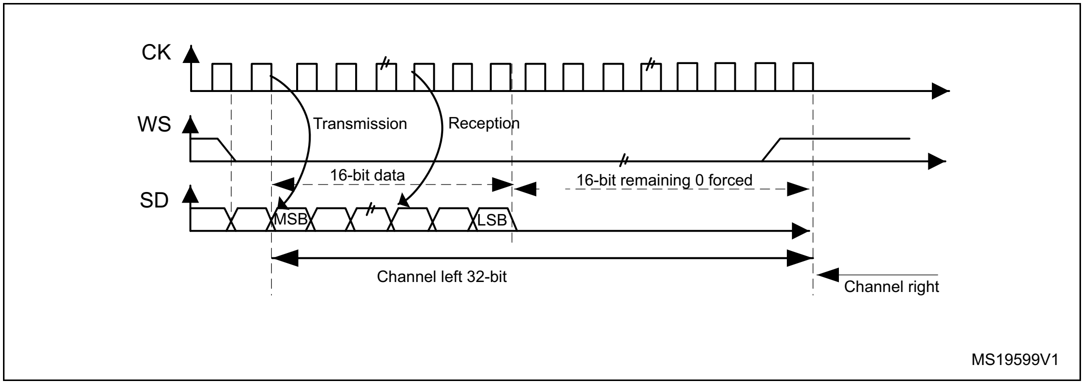
  If data 0x8EAA33 is received:

Figure 591. Receiving 0x8EAA33



Only the 8 MSB are sent to compare the 24 bits 8 LSBs have no meaning and can be anything

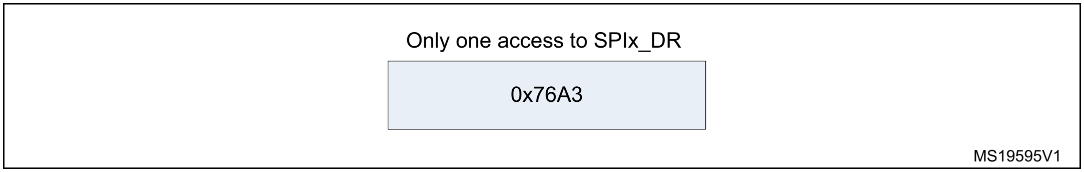
Figure 592. I^{2}S Philips standard (16-bit extended to 32-bit packet frame)



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 593* is required.

Figure 593. 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.

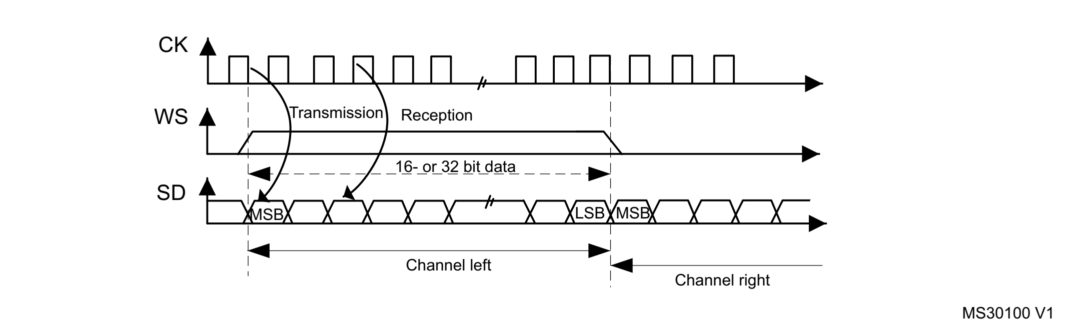
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

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 594. 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 595. MSB justified 24-bit frame length

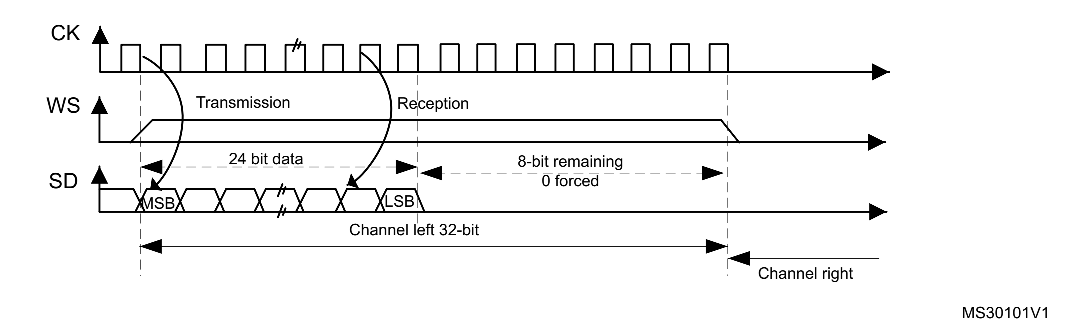
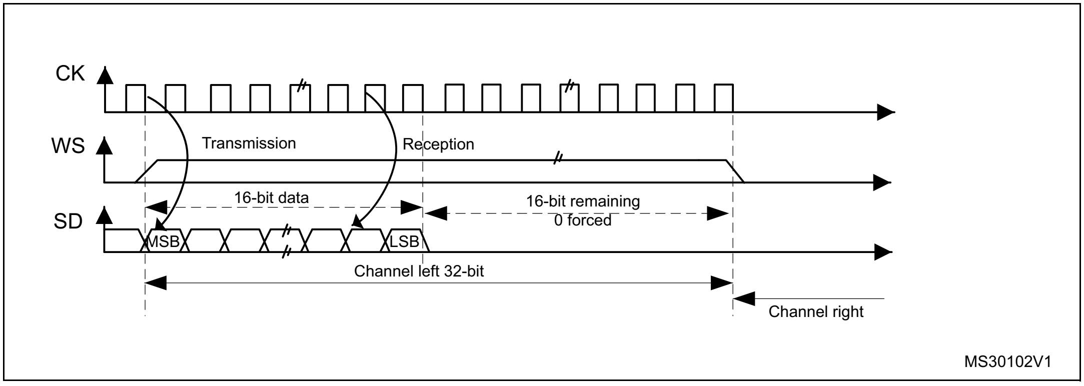


Figure 596. MSB justified 16-bit extended to 32-bit packet frame



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 597. LSB justified 16-bit or 32-bit full-accuracy

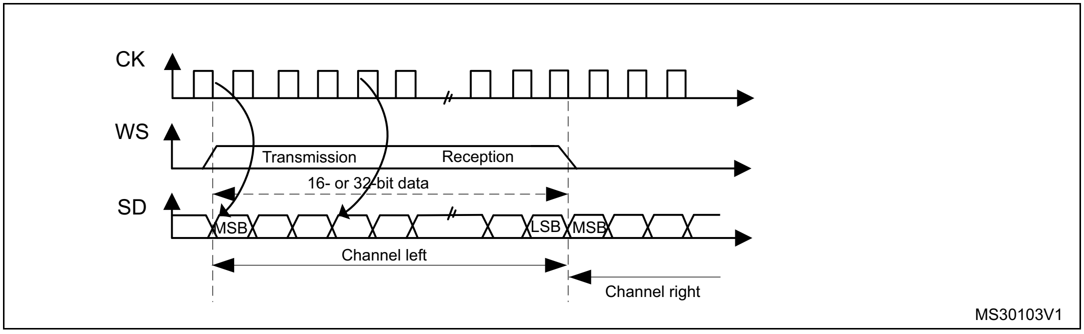
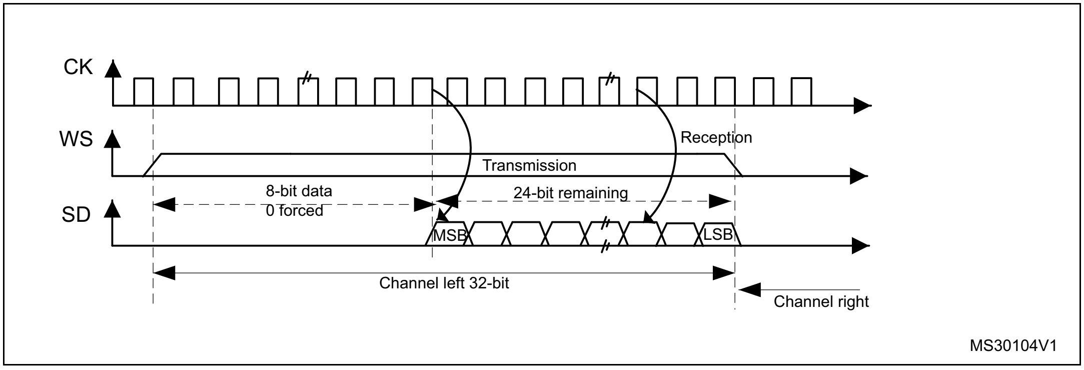


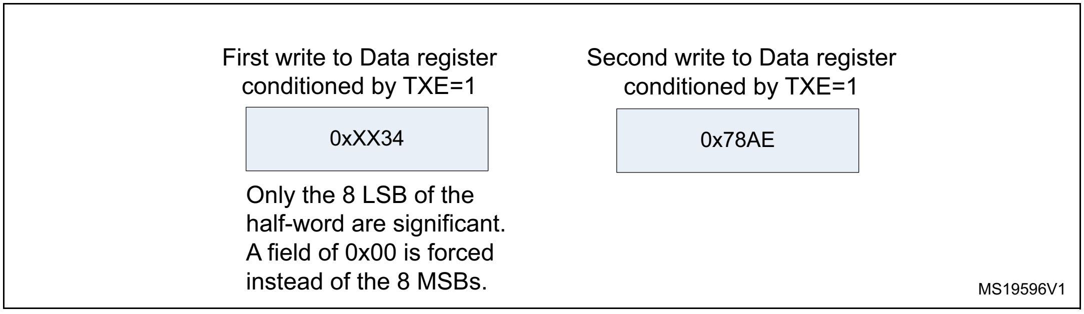
Figure 598. LSB justified 24-bit frame length



- 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 599. Operations required to transmit 0x3478AE


First write to Data register
conditioned by TXE=1
0xXX34
Only the 8 LSB of the
half-word are significant.
A field of 0x00 is forced
instead of the 8 MSBs.
Second write to Data register
conditioned by TXE=1
0x78AE
MS19596V1

- In reception mode:

  If data 0x3478AE are received, two successive read operations from the SPIx_DR register are required on each RXNE event.

Figure 600. Operations required to receive 0x3478AE

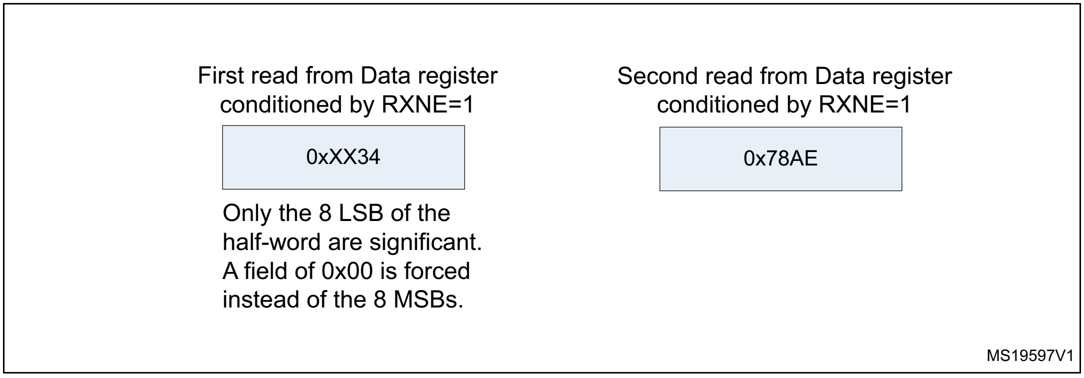
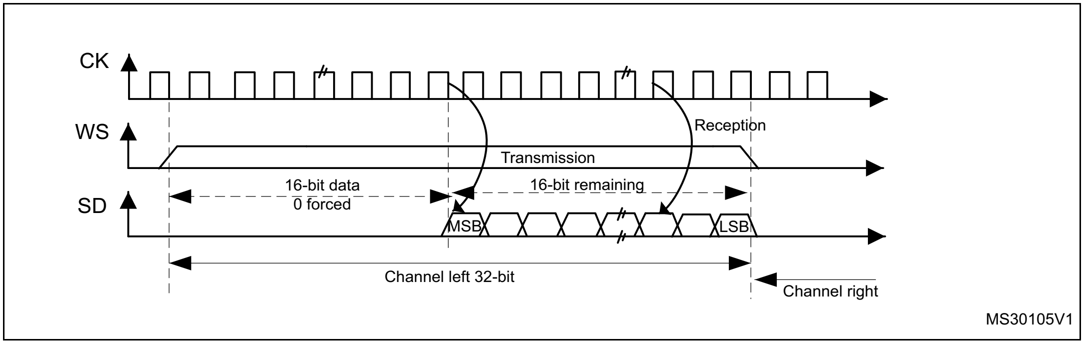

First read from Data register
conditioned by RXNE=1
0xXX34
Only the 8 LSB of the
half-word are significant.
A field of 0x00 is forced
instead of the 8 MSBs.
Second read from Data register
conditioned by RXNE=1
0x78AE
MS19597V1

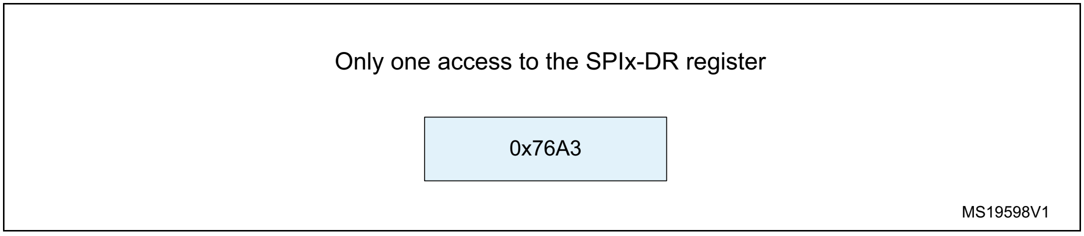
Figure 601. LSB justified 16-bit extended to 32-bit packet frame



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 602* is required.

Figure 602. 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.

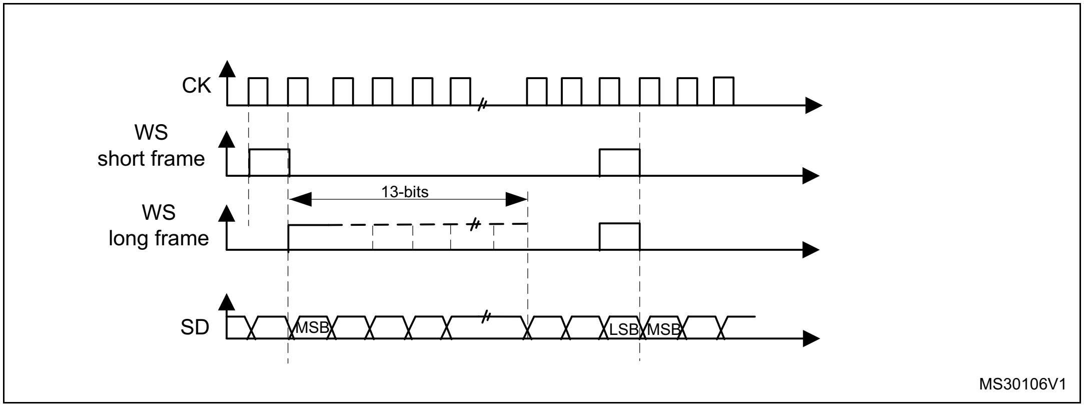
PCM standard

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.

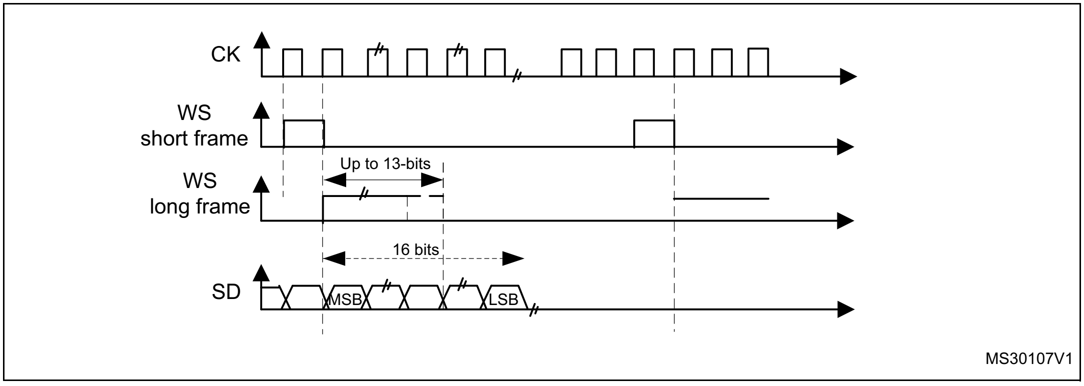
Figure 603. PCM standard waveforms (16-bit)



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 604. PCM standard waveforms (16-bit extended to 32-bit packet frame)

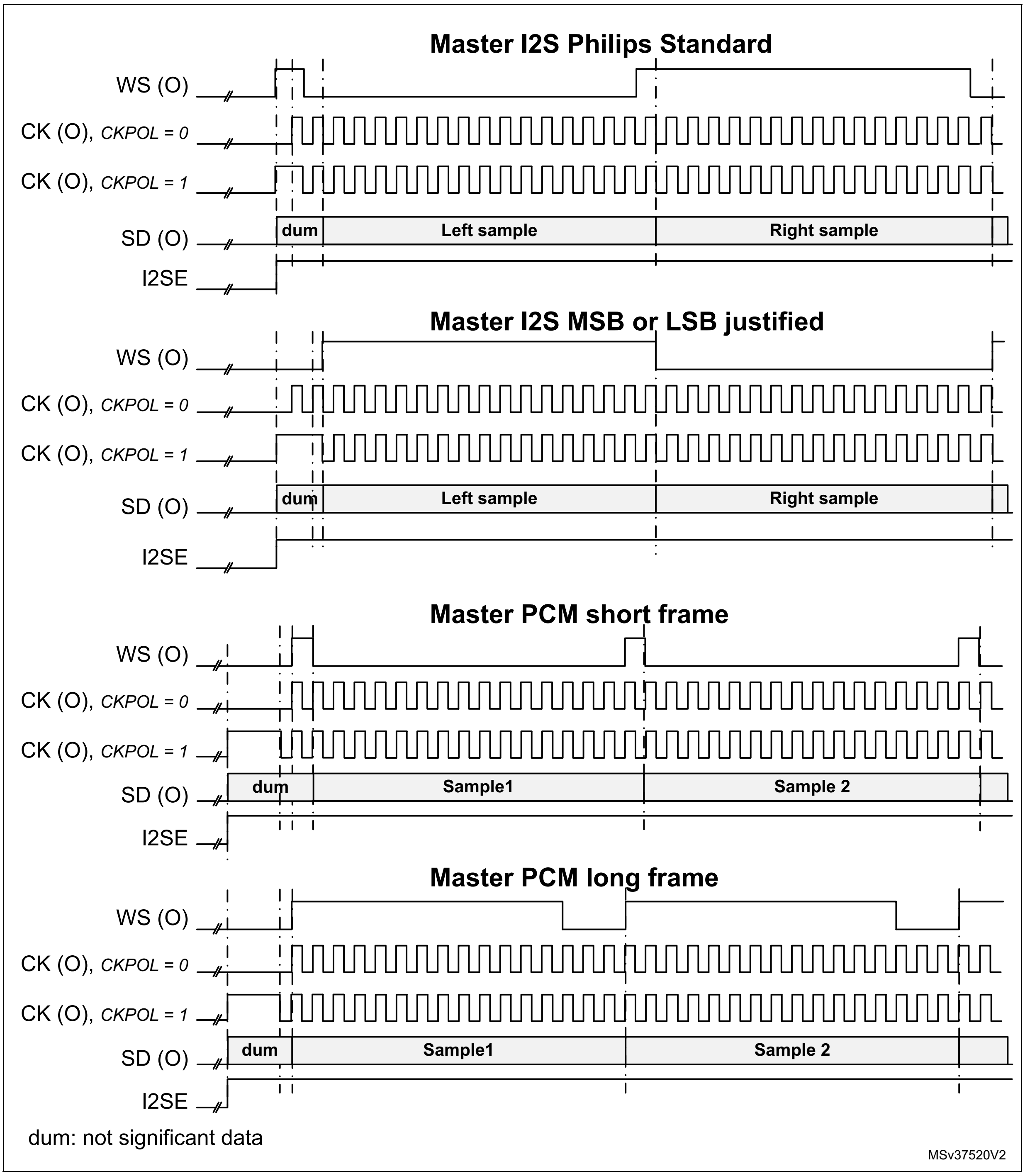


Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPIx_I2SCFGR register) even in slave mode.

39.7.3 Start-up description

The*Figure 605* 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 605. 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 I^2S 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 I^2S Philips standard, or when WS = 0 for other standards.

39.7.4 Clock generator

The I^2S bit rate determines the data flow on the I^2S data line and the I^2S clock signal frequency.

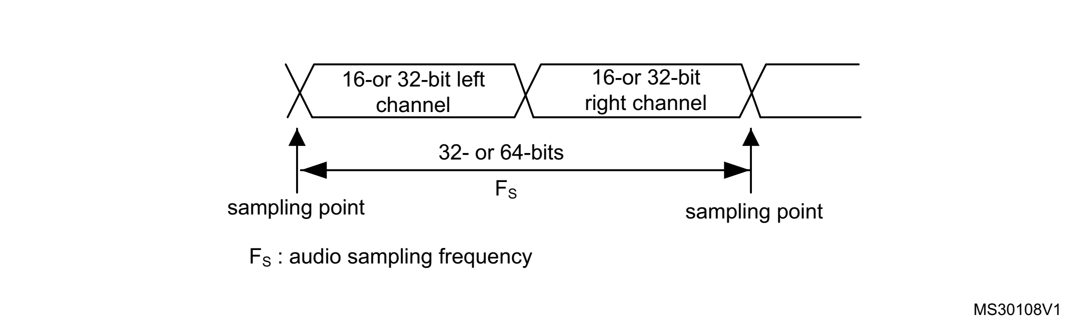
I^2S bit rate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I^2S bit rate is calculated as follows:

I^2S bit rate = 16 × 2 × f_S

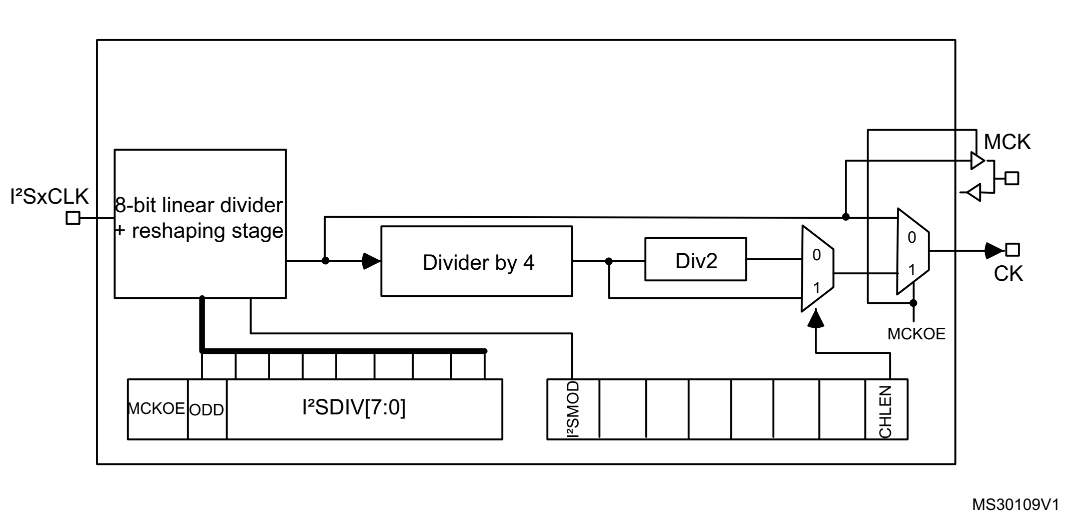
It is: I^2S bit rate = 32 x 2 x f_S if the packet length is 32-bit wide.

Figure 606. Audio sampling frequency definition


16-or 32-bit left channel
16-or 32-bit right channel
32- or 64-bits
F_S
sampling point
sampling point
F_S : audio sampling frequency
MS30108V1

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 607. I^2S clock generator architecture



1. Where x can be 2 or 3.

Figure 607 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 I^{2}S 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: *Note that I2SDIV must be strictly higher than 1.*

Table 361 provides example precision values for different clock configurations.

Note: *Other configurations are possible that allow optimum clock precision.*

Table 361. Audio-frequency precision using standard 8 MHz 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.

39.7.5 I^{2}S 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 39.7.4: Clock generator*).
3. Set the I2SMOD bit in the SPIx_I2SCFGR register to activate the I2S functions and choose the I^{2}S 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 I^{2}S 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 I^{2}S standard mode selected, refer to *Section 39.7.2: 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 39.7.5: I^{2}S 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 I^{2}S standard mode selected, refer to *Section 39.7.2: 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, I^{2}S 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.

39.7.6 I^{2}S 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 I^{2}S 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 39.7.2: 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 39.7.6: I^{2}S 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 I^{2}S standard mode selected, refer to *Section 39.7.2: 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.

39.7.7 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).

39.7.8 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^{2}S 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.

39.7.9 DMA features

In I^{2}S 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^{2}S mode since there is no data transfer protection system.

39.8 I2S interrupts

Table 362 provides the list of I2S interrupts.

Table 362. 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	

39.9 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.

39.9.1 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 1747.

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^{2}S 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^{2}S mode and SPI TI mode except the case when CRC is applied at TI mode.

39.9.2 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 1747 if the CRCEN bit is set.

This bit is not used in I^{2}S 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 1747 if the CRCEN bit is set.

This bit is not used in I^{2}S 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^{2}S 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^{2}S 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

39.9.3 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^{2}S 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 I^{2}S 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^{2}S slave mode. Refer to *Section 39.5.11: SPI error flags* and *Section 39.7.8: 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 39.5.10: SPI status flags and Procedure for disabling the SPI on page 1747.

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 1779* 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 1757* 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 1779* 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

39.9.4 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 39.5.9: 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.

39.9.5 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.

39.9.6 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.

39.9.7 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.

39.9.8 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*^{2}*S Philips Standard is used, or a rising edge for other standards.*

The appropriate ***level*** *is a low level on WS signal when I*^{2}*S Philips Standard is used, or a high level for other standards.*

Please refer to Section 39.7.3: 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^{2}S Philips standard

01: MSB justified standard (left justified)

10: LSB justified standard (right justified)

11: PCM standard

For more details on I^{2}S standards, refer to *Section 39.7.2 on page 1763*

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.

39.9.9 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 39.7.3 on page 1770*.

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 39.7.3 on page 1770*.

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.

39.9.10 SPI/I2S register map

Table 363 shows the SPI/I2S register map and reset values.

Table 363. 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 2.2 on page 81* for the register boundary addresses.

40 Serial audio interface (SAI)

40.1 Introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many stereo or mono audio applications may be targeted. I2S standards, LSB or MSB-justified, PCM/DSP, TDM, and AC’97 protocols may be addressed for example. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has it own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

40.2 SAI main features

- Two independent audio subblocks which can be transmitters or receivers with their respective FIFO.
- 8-word integrated FIFOs for each audio subblock.
- Synchronous or asynchronous mode between the audio subblocks.
- Master or slave configuration independent for both audio subblocks.
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode.
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit.
- Audio protocol: I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required.
- Up to 16 slots available with configurable size.
- Number of bits by frame can be configurable.
- Frame synchronization active level configurable (offset, bit length, level).
- First active bit position in the slot is configurable.
- LSB first or MSB first for data transfer.
- Mute mode.
- Stereo/Mono audio frame capability.
- Communication clock strobing edge configurable (SCK).
- Error flags with associated interrupts if enabled respectively.
  - Overrun and underrun detection,
  - Anticipated frame synchronization signal detection in slave mode,
  - Late frame synchronization signal detection in slave mode,
  - Codec not ready for the AC’97 mode in reception.
- Interrupt sources when enabled:
  - Errors,
  - FIFO requests.

- 2-channel DMA interface.

40.3 SAI implementation

Table 364. STM32G4 Series SAI features [(1)]

SAI features	SAI1
I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97	X
FIFO size	8 words
SPDIF	X
PDM	X[(2)]

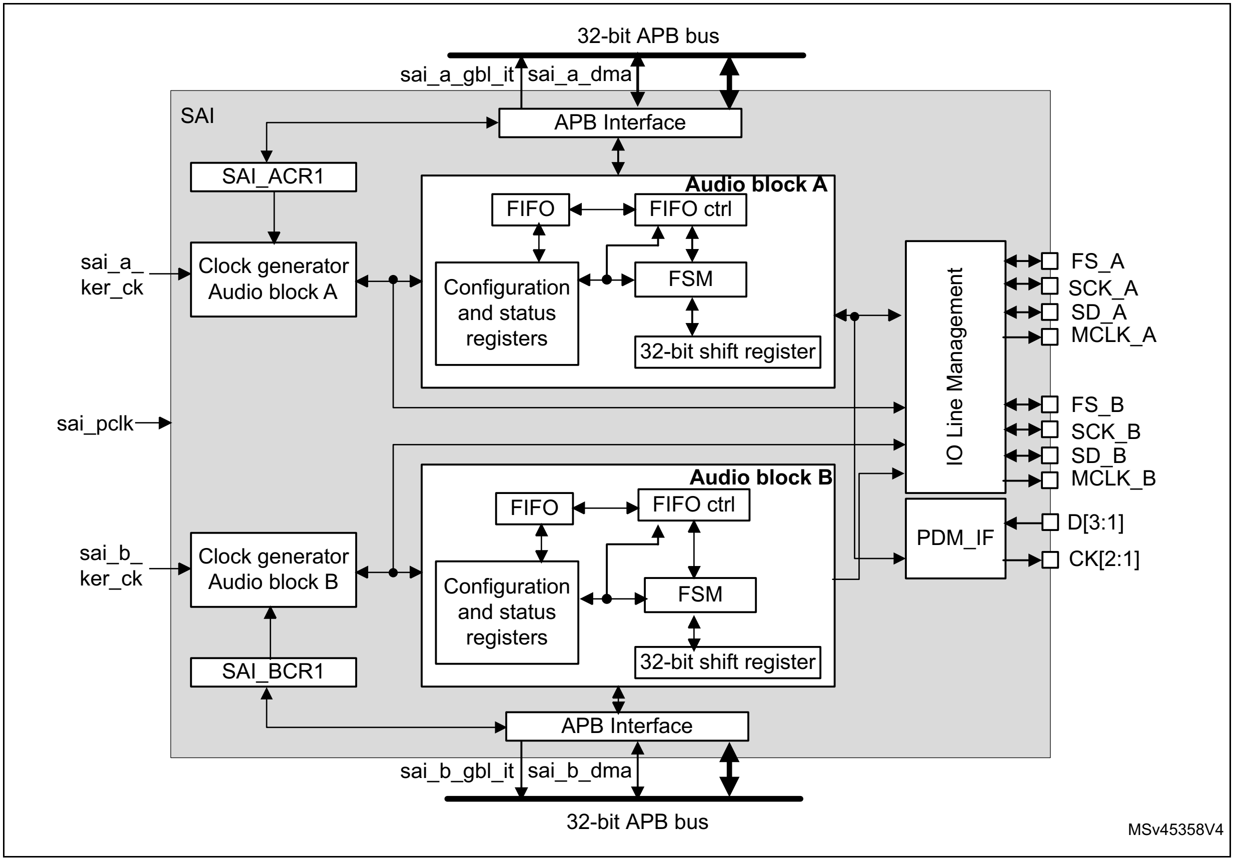
1. ‘X’ = supported, ‘-’ = not supported.
2. Only signals D[3:1], and CK[2:1] are available.

40.4 SAI functional description

40.4.1 SAI block diagram

Figure 608 shows the SAI block diagram while *Table 365* and *Table 366* list SAI internal and external signals.

Figure 608. SAI functional block diagram



The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA in order to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set-up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC’97 protocol, it is an SAI output even if the SAI (link controller) is set-up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where 'x' represents the SAI A or B subblock.

40.4.2 SAI pins and internal signals

Table 365. SAI internal input/output signals

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts.
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests.
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock.
sai_pclk	Input	APB clock.

Table 366. SAI input/output pins

Name	Signal type	Comments
SAI_SCK_A/B	Input/output	Audio block A/B bit clock.
SAI_MCLK_A/B	Output	Audio block A/B master clock.
SAI_SD_A/B	Input/output	Data line for block A/B.
SAI_FS_A/B	Input/output	Frame synchronization line for audio block A/B.
SAI_CK[4:1]	Output	PDM bitstream clock[1].
SAI_D[4:1]	Input	PDM bitstream data[1].

[1] These signals might not be available in all SAI instances. Please refer to *Section 40.3: SAI implementation* for details.

40.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.

Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI are enabled by SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock line, data line and synchronization line in slave mode.

In master TX mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO, However FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave TX mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold (FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

40.4.4 SAI synchronization mode

SAI sub-clock A and B can be synchronized.

Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

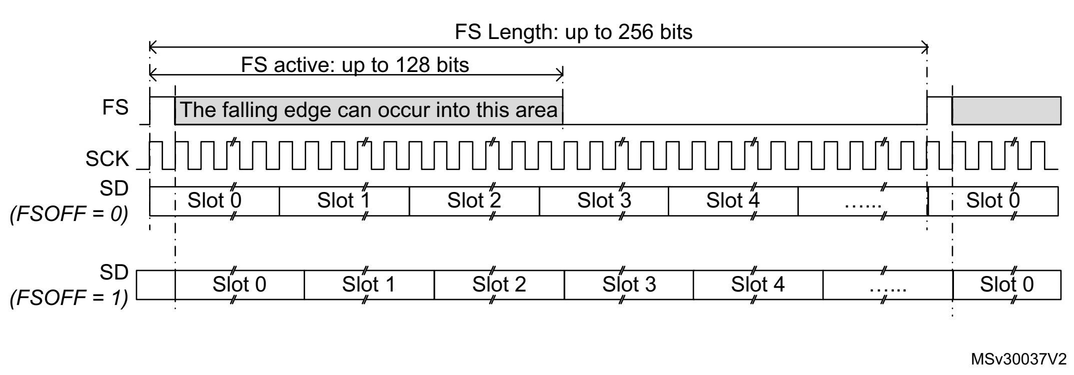
40.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24 or 32 bits. During the transfer, either the MSB or the LSB of the data are sent first, depending on the configuration of bit LSBFIRST in the SAI_xCR1 register.

40.4.6 Frame synchronization

The FS signal acts as the Frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable in order to target the different audio protocols with their own specificities concerning this Frame synchronization behavior. This reconfigurability is done using register SAI_xFRCR. *Figure 609* illustrates this flexibility.

Figure 609. Audio frame



In AC’97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC’97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode

  The audio frame length can be configured to up to 256 bit clock cycles, by setting FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit is extended to 0 or the SD line is released to HI-z depending the state of bit TRIS in the SAI_xCR2 register (refer to *FS signal role*). In reception mode, the remaining bit is ignored.

If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to *Section 40.4.8: SAI clock generator*".

- Slave mode

  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the Frame synchronization signal during an on-going audio frame. In this case an error is generated. For more details refer to *Section 40.4.14: Error flags*.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame is 8.

Frame synchronization polarity

FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. Start of frame is synchronized to this signal. It is effective only if the start of frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to *Section 40.4.14: Error flags*).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in *Section 40.4.14: Error flags*), but the audio communication flow is not interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register allow configuring the length of the active level of the Frame synchronization signal. The length can be set from 1 to 128 bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the Frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). FSOFF bit in the SAI_xFRCR register allows to choose one of the two configurations.

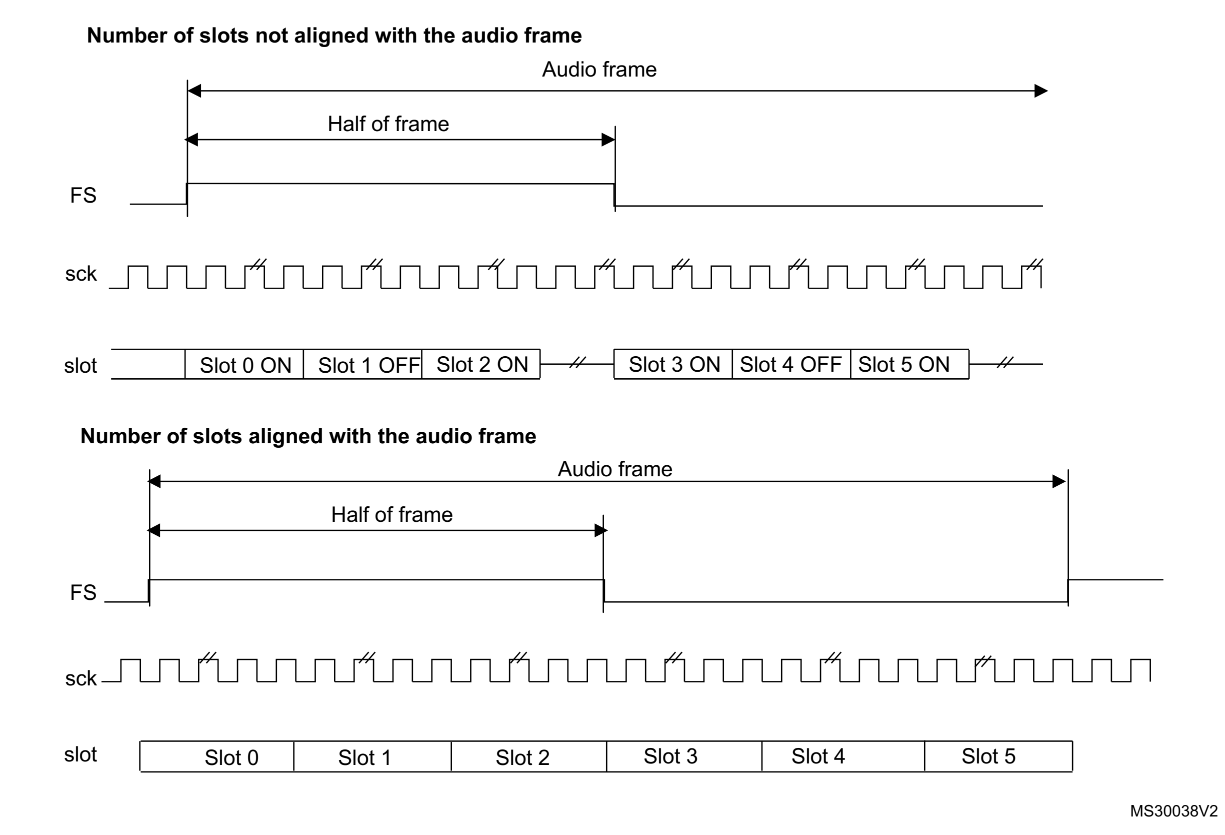
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

- 0: start of frame, like for instance the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S, the MSB or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise if TRIS = 1, the SD line is released to HI-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 610. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

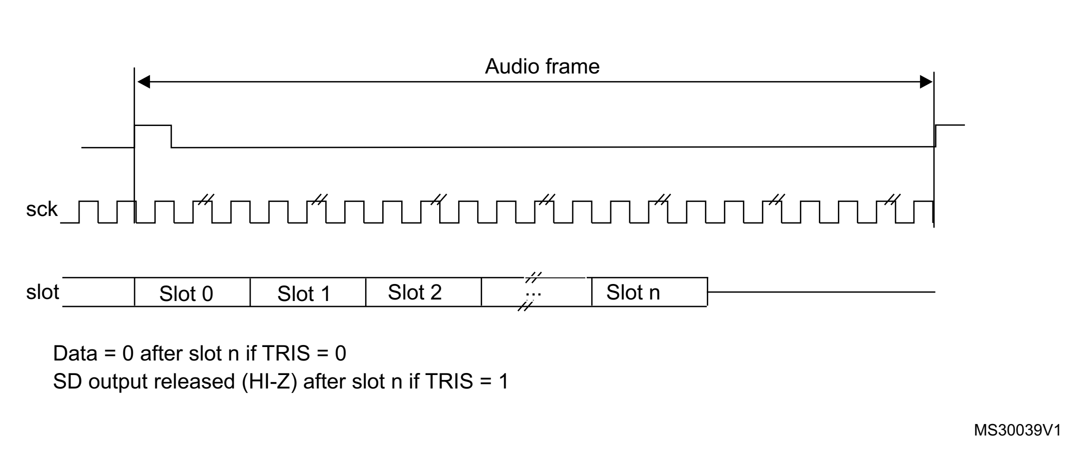


1. The frame length should be even.

If FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:

- if TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
- if TRIS = 1, the line is released to HI-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 611. FS role is start of frame (FSDEF = 0)


Audio frame
sck
slot
Slot 0
Slot 1
Slot 2
...
Slot n
Data = 0 after slot n if TRIS = 0
SD output released (HI-Z) after slot n if TRIS = 1
MS30039V1

The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

40.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

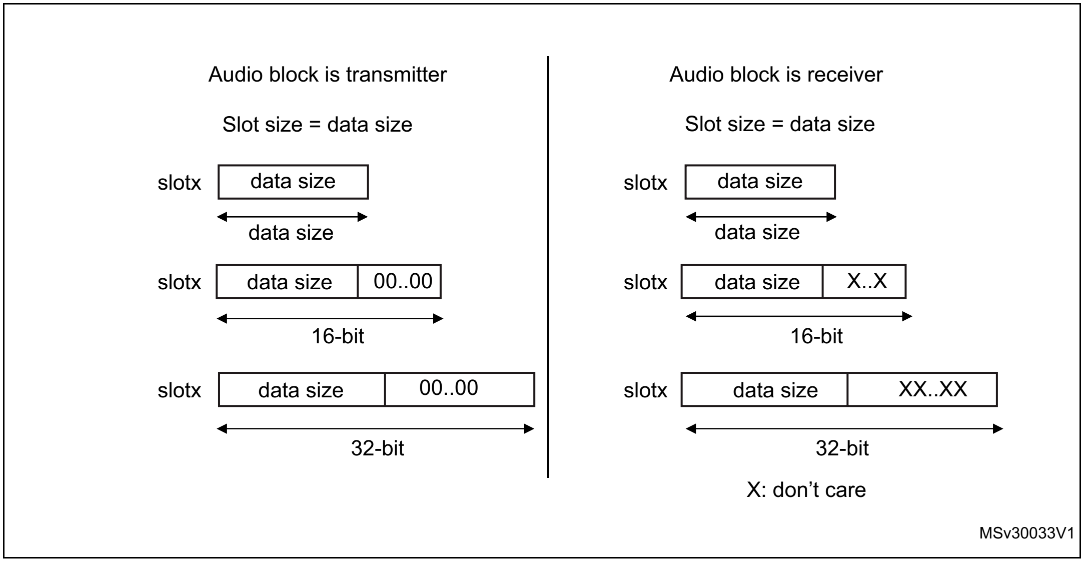
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When a invalid slot is transferred, the SD data line is either forced to 0 or released to HI-z depending on TRIS bit configuration (refer to *Output data line management on an inactive slot*) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

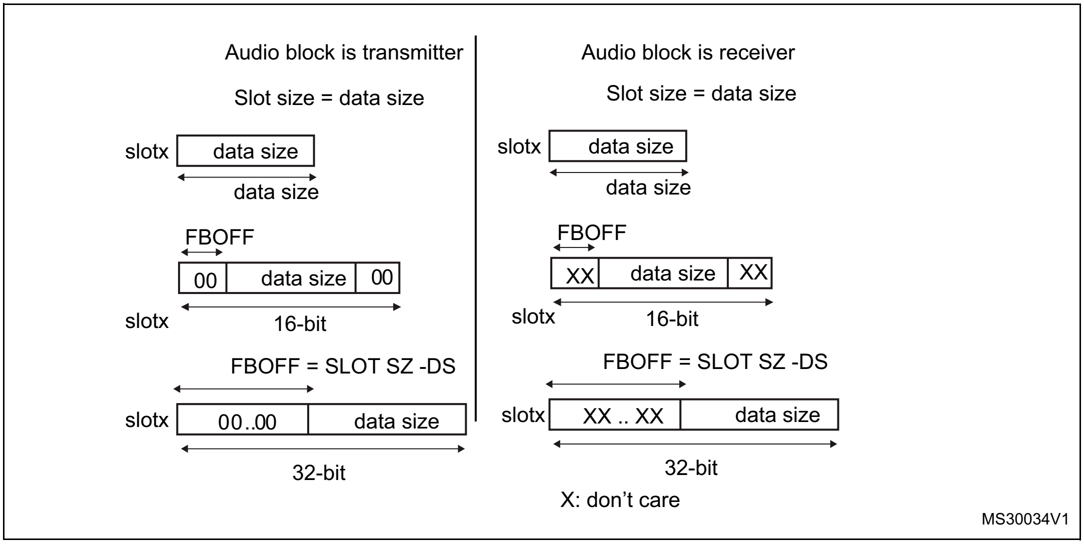
The slot size is also configurable as shown in *Figure 612.* The size of the slots is selected by setting SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 612. Slot size configuration with FBOFF = 0 in SAI_xSLOTR



It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 613. First bit offset



It is mandatory to respect the following conditions to avoid bad SAI behavior:

FBOFF ≤(SLOTSZ - DS),

DS ≤SLOTSZ,

NBSLOT x SLOTSZ ≤FRL (frame length),

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC’97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in *Section 40.4.11: AC’97 link controller*.

40.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

Generation of the master clock (MCLK_x)

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it allows generating the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI guarantees that there is no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via MCKEN bit (with SAIEN = 0).

Table 367 shows MCLK_x activation conditions.

Table 367. MCLK_x activation conditions

MCLKEN	NODIV	SAIEN for block x	MCLK_x
0	X	0	**Disabled**
1			Enabled
0	1	1	**Disabled**
1			Enabled
X	0		Enabled

Note: MCLK_x can also be generated in AC’97 mode, when MCLKEN is set to 1.

Generation of the bit clock (SCK_x)

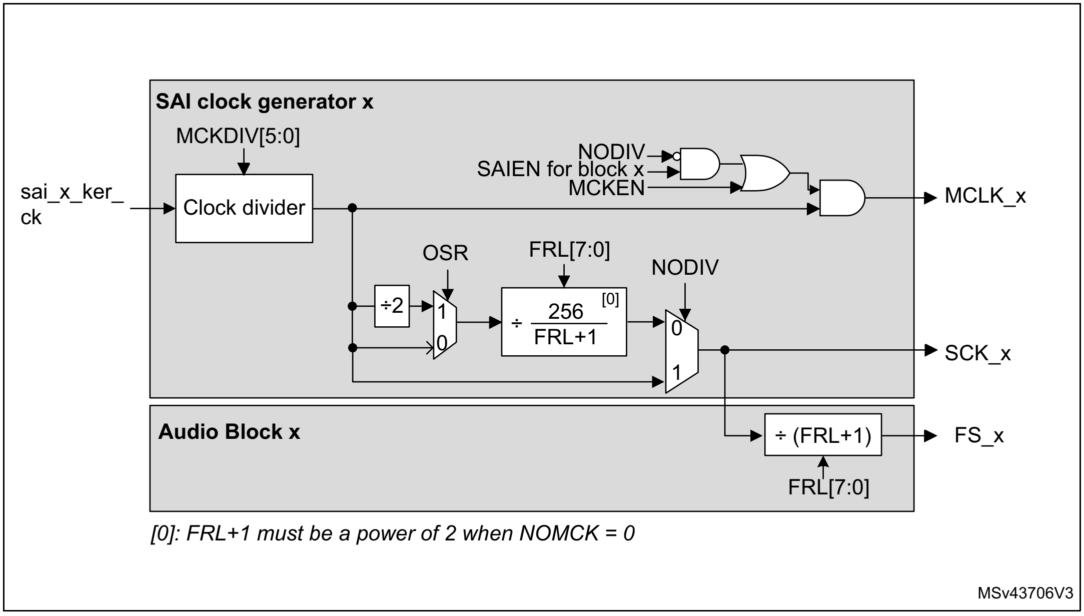
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

Figure 614 illustrates the architecture of the audio block clock generator.

Figure 614. Audio block clock generator overview



The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given hereafter.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (i.e. FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.

Clock generator programming when NODIV = 0

In that case, MCLK_xfrequency is:

- F_{MCLK_x} = 256 x F_{FS_x} if OSR = 0
- F_{MCLK_x} = 512 x F_{FS_x} if OSR = 1

When MCKDIV is different from 0, MCLK_x frequency is given by the formula below:

$$F_{MCLK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The bit clock frequency (SCK_x) is given by the following formula:

$$F_{SCK\_x} = \frac{F_{sai\_x\_ker\_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}$$

Note: When NODIV is equal to 0, (FRL+1) must be a power of two. In addition (FRL+1) must range between 8 and 256. (FRL +1) represents the number of bit clock in the audio frame.

When MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if (FRL+1) = 2^{8}.

It is recommended, to program MCKDIV to an even value or to big values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

$$F_{SCK\_x} = F_{MCLK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS_x) in given by the following formula:

$$F_{FS\_x} = \frac{F_{sai\_x\_ker\_ck}}{(FRL + 1) \times MCKDIV}$$

Note: When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming examples

Table 368 gives programming examples for 48, 96 and 192 kHz.

Table 368. Clock generator programming examples

Input sai_x_ker_ck clock frequency	MCLK	F_{MCLK}/F_{FS}	FRL [(1)]	OSR	NODIV	MCKEN	MCKDIV[5:0]	Audio Sampling frequency (F_{FS})
98.304 MHz	Y	512	2^N-1	1	0	1	0 or 1	192 kHz
		512	2^N-1	1	0	1	2	96 kHz
		512	2^N-1	1	0	1	4	48 kHz
		256	2^N-1	0	0	1	2	192 kHz
		256	2^N-1	0	0	1	4	96 kHz
		256	2^N-1	0	0	1	8	48 kHz
	N	-	63	-	1	0	8	192 kHz
		-	63	-	1	0	16	96 kHz
		-	63	-	1	0	32	48 kHz

1. N is an integer value between 3 and 8.

40.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. There is therefore only one FIFO request linked to FREQ bit in the SAI_xSR register.

An interrupt is generated if FREQIE bit is enabled in the SAI_xIM register. This depends on:

- FIFO threshold setting (FLVL bits in SAI_xCR2)
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in SAI_xSR register) if no data are available in SAI_xDR register (FLVL[2:0] bits in SAI_xSR is less than 001b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is no more empty (FLVL[2:0] bits in SAI_xSR are different from 0b000) i.e one or more data are stored in the FIFO.
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit set by hardware to 1 in SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher or equal to 0b010).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in

SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 011b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher or equal to 011b).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 011b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if less than three quarters of the FIFO contain data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contain data (FLVL[2:0] bits in SAI_xSR are higher or equal to 0b100).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR is less than 101b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR is equal to 101b value).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least one data is available in SAI_xDR register(FLVL[2:0] bits in SAI_xSR is higher or equal to 001b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in SAI_xSR is equal to 0b000) i.e no data are stored in FIFO.
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO quarter fully (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 0b010). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR is less than 0b010).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO half fully (FTH[2:0] set to 0b010 value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 011b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR is less than 011b).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO three quarter full(FTH[2:0] set to 011b value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 0b100). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations avalable(FLVL[2:0] bits in SAI_xSR is less than 0b100).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO full(FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR is equal to 101b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR is less than 101b).

Like interrupt generation, the SAI can use the DMA if DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data should be right aligned when it is written in the SAI_xDR.

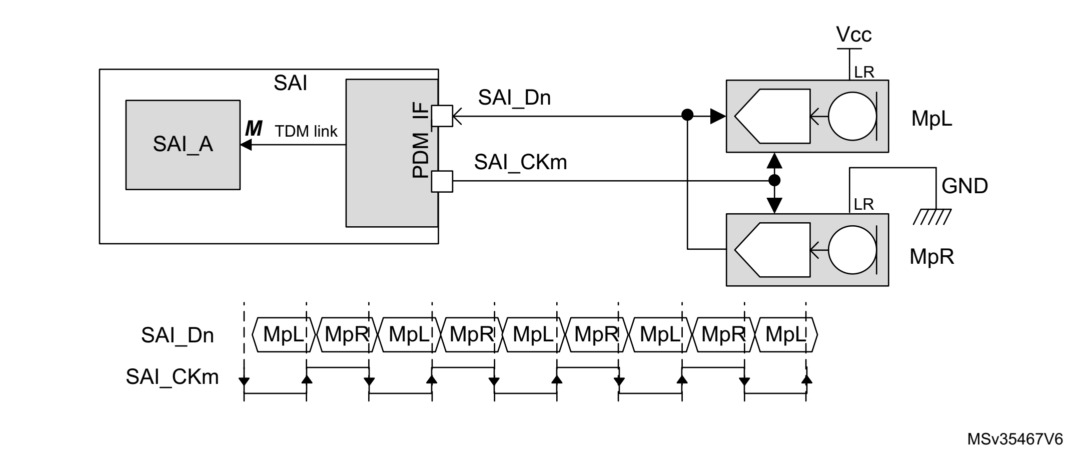
Data received are right aligned in the SAI_xDR.

The FIFO pointers can be reinitialized when the SAI is disabled by setting bit FFLUSH in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled the data present in the FIFO are lost automatically.

40.4.10 PDM Interface

The PDM (Pulse Density Modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. *Figure 615* shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 615. PDM typical connection and timing



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the TDM interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).

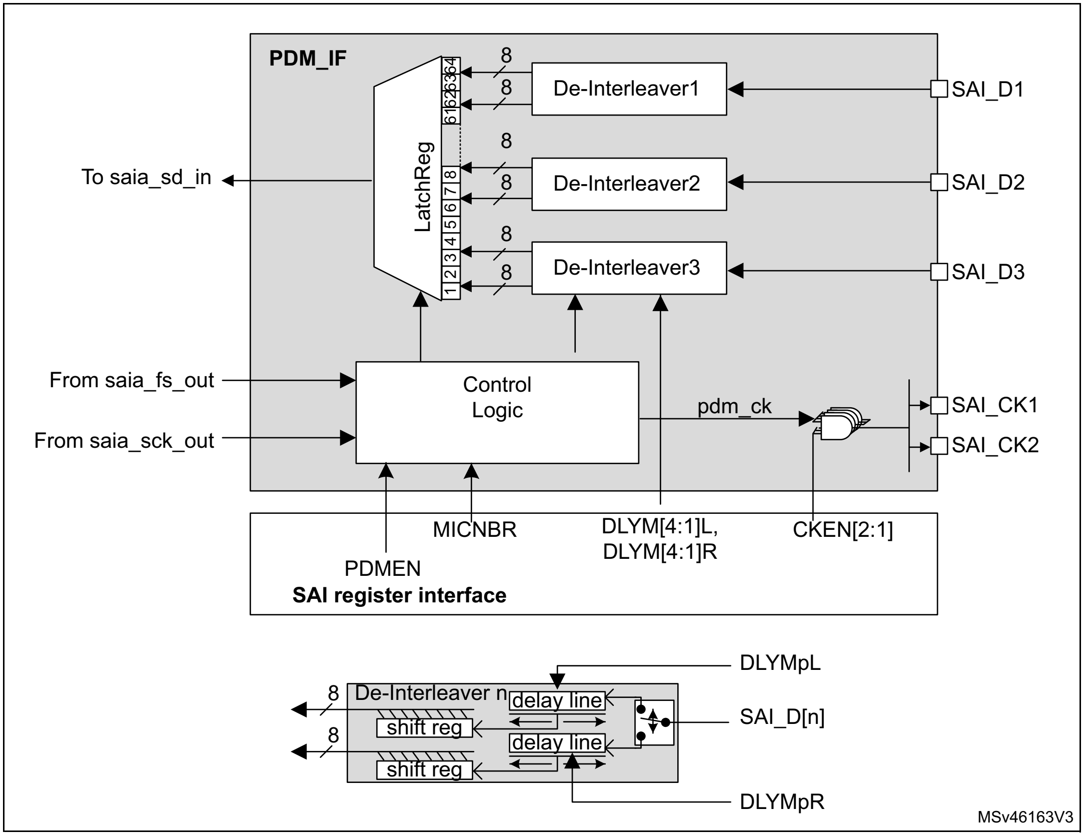
The data processing sequence into the PDM is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the TDM interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line in order to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A via the serial data line of the TDM interface.

Figure 616 hereafter shows the block diagram of PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface *does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.*

Figure 616. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM interface can be enabled through the PDMEN bit in SAI_PDMCR register. However the PDM interface must be enabled prior to enabling SAI_A block.

To reduce the memory footprint, the user can select the amount of microphones the application needs. This can be done through MICNBR[1:0] bits. It is possible to select between 2,4,6 or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

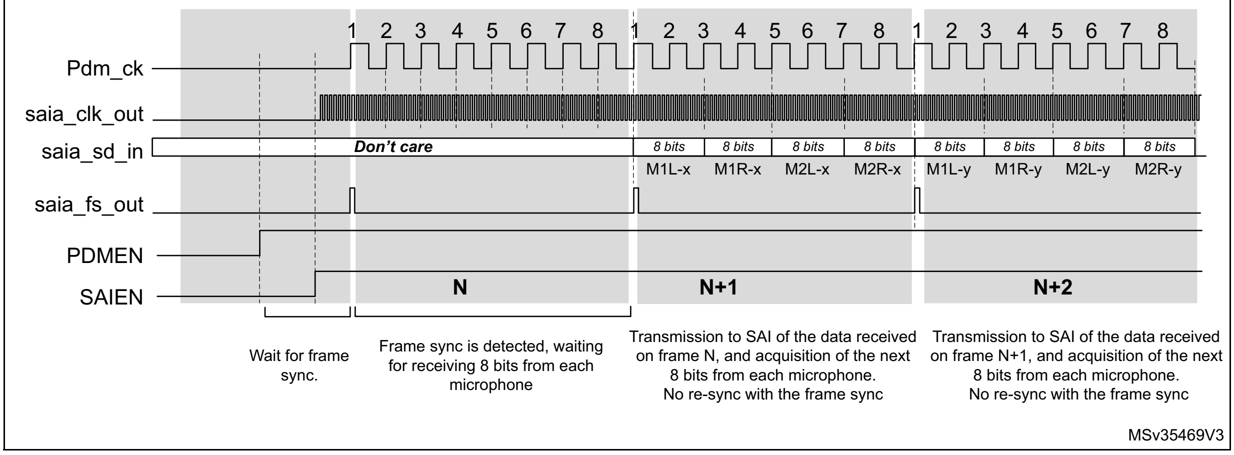
1. Configure SAI_A in TDM master mode (see *Table 369*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first 2 TDMA frames received on SAI_ADR are invalid and must be dropped.

Start-up sequence

Figure 617 shows the start-up sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the TDM interface.

Figure 617. Start-up sequence


Pdm_ck
saia_clk_out
saia_sd_in
saia_fs_out
PDMEN
SAIEN
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
Don't care
8 bits
M1L-x
M1R-x
M2L-x
M2R-x
M1L-y
M1R-y
M2L-y
M2R-y
N
N+1
N+2
Wait for frame sync.
Frame sync is detected, waiting for receiving 8 bits from each microphone
Transmission to SAI of the data received on frame N, and acquisition of the next 8 bits from each microphone. No re-sync with the frame sync
Transmission to SAI of the data received on frame N+1, and acquisition of the next 8 bits from each microphone. No re-sync with the frame sync
MSv35469V3

SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- The amount of microphones
- The slot width selected
- LSBFIRST bit.

The slot width defines the amount of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the amount of words stored into the memory. However the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

In the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contain 8 useful bits. This increases the amount of words stored into the memory. However, it offers the advantage to avoid extra processing since each word contains information from one microphone.

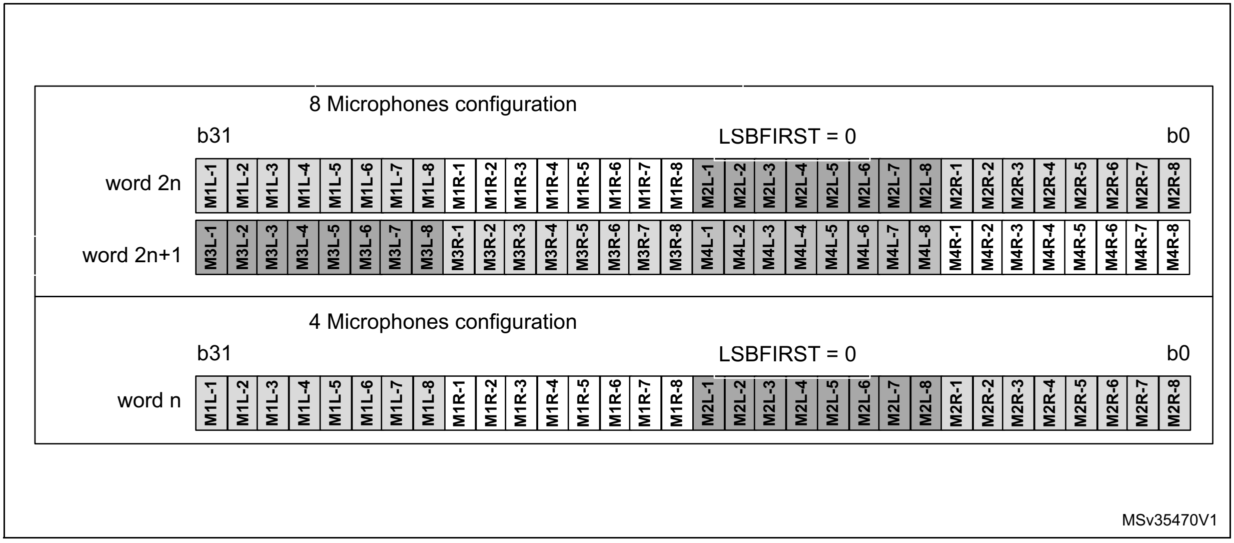
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 618*.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

Figure 618. SAI_ADR format in TDM, 32-bit slot width



8 Microphones configuration

word 2n	M1L-1	M1L-2	M1L-3	M1L-4	M1L-5	M1L-6	M1L-7	M1L-8	M1R-1	M1R-2	M1R-3	M1R-4	M1R-5	M1R-6	M1R-7	M1R-8	M2L-1	M2L-2	M2L-3	M2L-4	M2L-5	M2L-6	M2L-7	M2L-8	M2R-1	M2R-2	M2R-3	M2R-4	M2R-5	M2R-6	M2R-7	M2R-8
word 2n+1	M3L-1	M3L-2	M3L-3	M3L-4	M3L-5	M3L-6	M3L-7	M3L-8	M3R-1	M3R-2	M3R-3	M3R-4	M3R-5	M3R-6	M3R-7	M3R-8	M4L-1	M4L-2	M4L-3	M4L-4	M4L-5	M4L-6	M4L-7	M4L-8	M4R-1	M4R-2	M4R-3	M4R-4	M4R-5	M4R-6	M4R-7	M4R-8

4 Microphones configuration

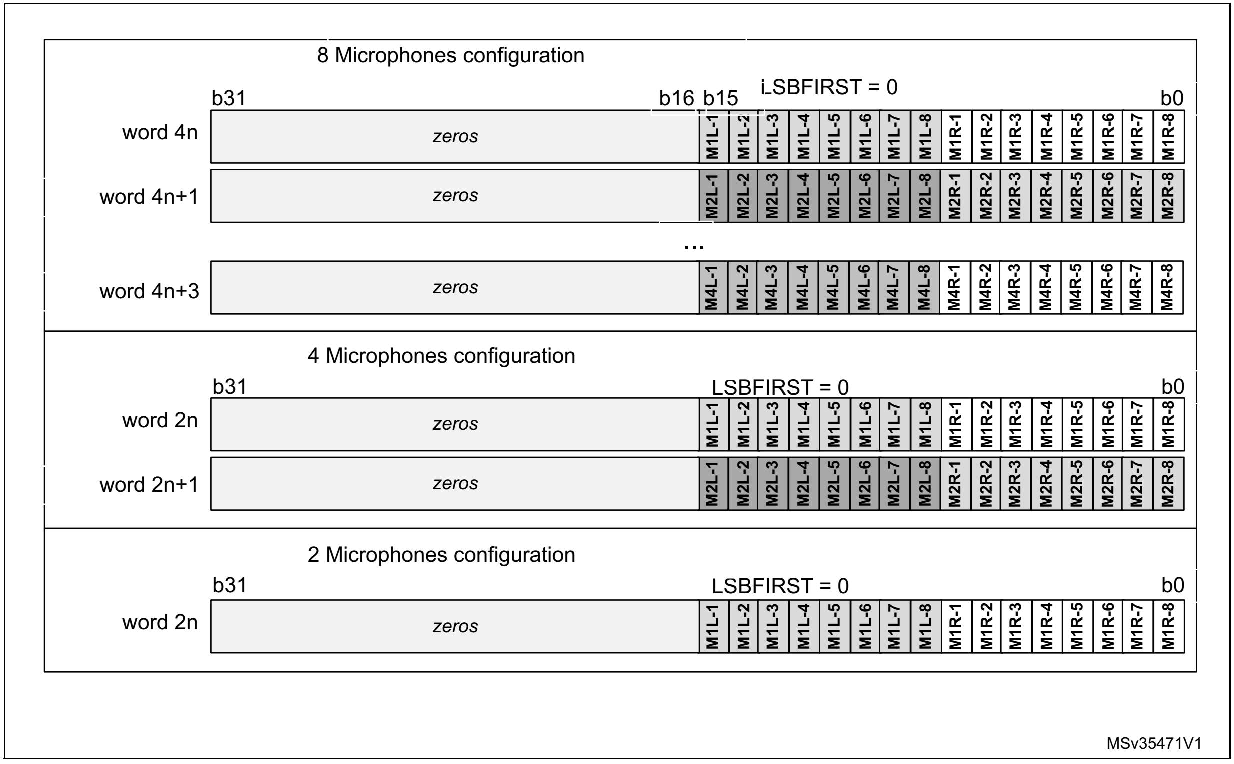
word n	M1L-1	M1L-2	M1L-3	M1L-4	M1L-5	M1L-6	M1L-7	M1L-8	M1R-1	M1R-2	M1R-3	M1R-4	M1R-5	M1R-6	M1R-7	M1R-8	M2L-1	M2L-2	M2L-3	M2L-4	M2L-5	M2L-6	M2L-7	M2L-8	M2R-1	M2R-2	M2R-3	M2R-4	M2R-5	M2R-6	M2R-7	M2R-8

- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 619*.

  For an 8 microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right aligned.

  For 4 or 2 microphone configuration, the SAI behavior is similar to 8-microphone configurations. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 619. SAI_ADR format in TDM, 16-bit slot width

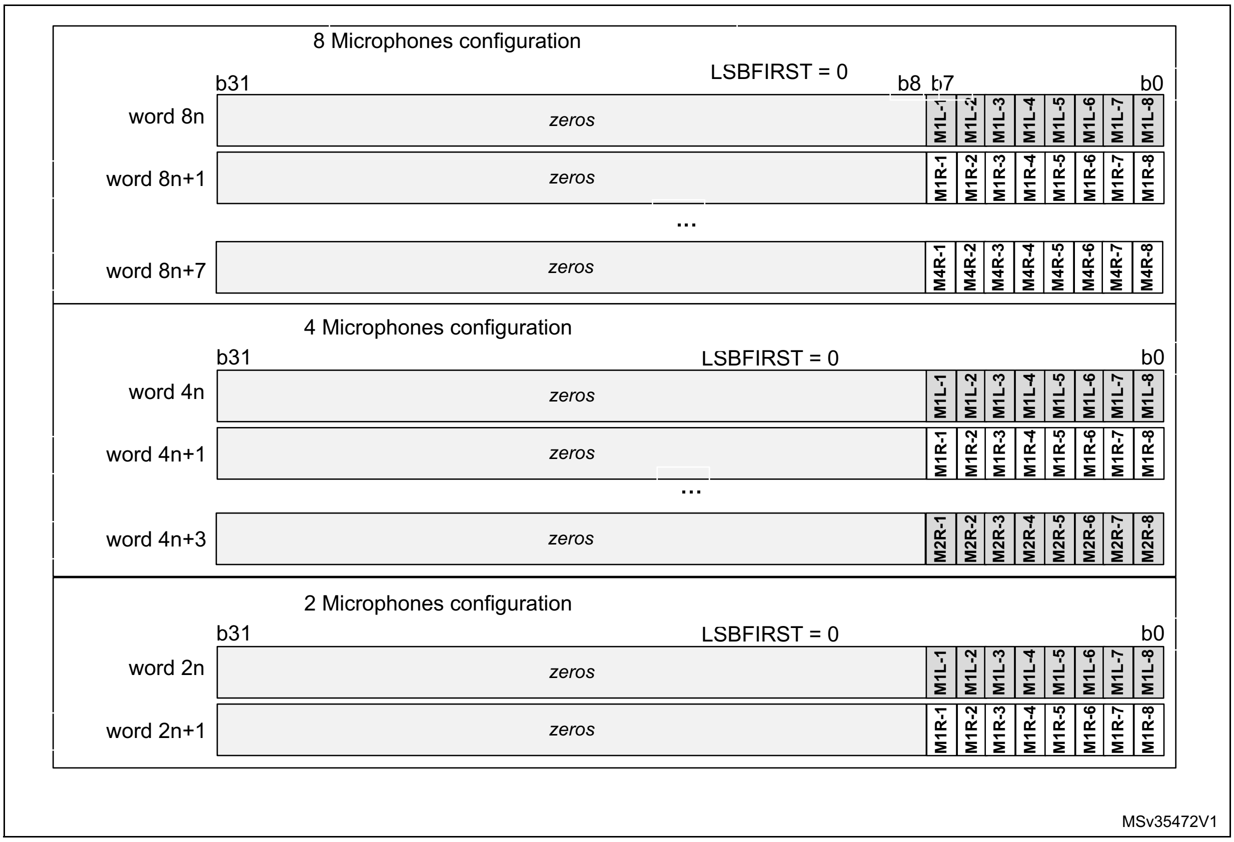

8 Microphones configuration
b31
b16 b15
LSBFIRST = 0
b0
word 4n
zeros
M1L-1 M1L-2 M1L-3 M1L-4 M1L-5 M1L-6 M1L-7 M1L-8 M1R-1 M1R-2 M1R-3 M1R-4 M1R-5 M1R-6 M1R-7 M1R-8
word 4n+1
zeros
M2L-1 M2L-2 M2L-3 M2L-4 M2L-5 M2L-6 M2L-7 M2L-8 M2R-1 M2R-2 M2R-3 M2R-4 M2R-5 M2R-6 M2R-7 M2R-8
...
word 4n+3
zeros
M4L-1 M4L-2 M4L-3 M4L-4 M4L-5 M4L-6 M4L-7 M4L-8 M4R-1 M4R-2 M4R-3 M4R-4 M4R-5 M4R-6 M4R-7 M4R-8
4 Microphones configuration
b31
LSBFIRST = 0
b0
word 2n
zeros
M1L-1 M1L-2 M1L-3 M1L-4 M1L-5 M1L-6 M1L-7 M1L-8 M1R-1 M1R-2 M1R-3 M1R-4 M1R-5 M1R-6 M1R-7 M1R-8
word 2n+1
zeros
M2L-1 M2L-2 M2L-3 M2L-4 M2L-5 M2L-6 M2L-7 M2L-8 M2R-1 M2R-2 M2R-3 M2R-4 M2R-5 M2R-6 M2R-7 M2R-8
2 Microphones configuration
b31
LSBFIRST = 0
b0
word 2n
zeros
M1L-1 M1L-2 M1L-3 M1L-4 M1L-5 M1L-6 M1L-7 M1L-8 M1R-1 M1R-2 M1R-3 M1R-4 M1R-5 M1R-6 M1R-7 M1R-8
MSv35471V1

- **Using a 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 620*.

  For an 8 microphone configuration, 8 consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right aligned.

  For 4 or 2 microphone configuration, the SAI behavior is similar to 8 microphone configurations. Up to 4 words of 8 bits are required to acquire a byte from 4 microphones and 2 words from 2 microphones.

Figure 620. SAI_ADR format in TDM, 8-bit slot width


8 Microphones configuration
LSBFIRST = 0
b31
b8
b7
b0
word 8n
zeros
M1L-1 M1L-2 M1L-3 M1L-4 M1L-5 M1L-6 M1L-7 M1L-8
word 8n+1
zeros
M1R-1 M1R-2 M1R-3 M1R-4 M1R-5 M1R-6 M1R-7 M1R-8
...
word 8n+7
zeros
M4R-1 M4R-2 M4R-3 M4R-4 M4R-5 M4R-6 M4R-7 M4R-8
4 Microphones configuration
b31
LSBFIRST = 0
b0
word 4n
zeros
M1L-1 M1L-2 M1L-3 M1L-4 M1L-5 M1L-6 M1L-7 M1L-8
word 4n+1
zeros
M1R-1 M1R-2 M1R-3 M1R-4 M1R-5 M1R-6 M1R-7 M1R-8
...
word 4n+3
zeros
M2R-1 M2R-2 M2R-3 M2R-4 M2R-5 M2R-6 M2R-7 M2R-8
2 Microphones configuration
b31
LSBFIRST = 0
b0
word 2n
zeros
M1L-1 M1L-2 M1L-3 M1L-4 M1L-5 M1L-6 M1L-7 M1L-8
word 2n+1
zeros
M1R-1 M1R-2 M1R-3 M1R-4 M1R-5 M1R-6 M1R-7 M1R-8
MSv35472V1

TDM configuration for PDM interface

SAI_A TDM interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the PDM interface as shown in *Table 369* to ensure a good connection with the PDM interface.

Table 369. TDM settings

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver
PRTCFG	0b00	Free protocol for TDM
DS	X	To be adjusted according to the required data format, in accordance to the frame length and the number of slots (FRL and NBSLOT). See *Table 370*.
LSBFIRST	X	This parameter can be used according to the wanted data format
CKSTR	0	Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.
MONO	0	Stereo mode
FRL	X	To be adjusted according to the number of microphones (MICNBR). See *Table 370*.
FSALL	0	Pulse width is one bit clock cycle
FSDEF	0	FS signal is a start of frame

Table 369. TDM settings (continued)

Bit Fields	Values	Comments
FSPOL	1	FS is active High
FSOFF	0	FS is asserted on the first bit of slot 0
FBOFF	0	No offset on slot
SLOTSZ	0	Slot size = data size
NBSLOT	X	To be adjusted according to the required data format, in accordance to the slot size, and the frame length (FRL and DS). See *Table 370*.
SLOTEN	X	To be adjusted according to NBSLOT
NODIV	1	No need to generate a master clock MCLK
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See *Table 370*.

Adjusting the bitstream clock rate

To properly program the SAI TDM interface, the user application must take into account the settings given in *Table 369*, and follow the below sequence:

1. Adjust the bit clock frequency (F_{SCK_A}) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK\_A} = F_{PDM\_CK} \times (MICNBR + 1) \times 2$$

   MICNBR can be 0,1,2 or 3 (0 = 2 microphones., see *Section 40.6.17)*

2. Set the frame length (FRL) using the following formula

$$FRL = (16 \times (MICNBR + 1)) - 1$$

3. Configure the slot size (DS) to a multiple of (FRL+1).

Table 370. Allowed TDM frame configuration[1]

Microphone Sample rate	Nber of Mic	Wanted SAI_CKn frequency	bit clock (SCK_A) frequency	Frame sync. (FS_A) frequency	FRL	DS	NBSLOT	Comments
48 kHz	up to 8	3.072 MHz	24.576 MHz	384 kHz	63	0b111	1	2 slots of 32 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b100	3	4 slots of 16 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	3.072 MHz	18.432 MHz	384 kHz	47	0b110	1	2 slots of 24 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b100	2	3 slots of 16 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	3.072 MHz	12.288 MHz	384 kHz	31	0b111	0	1 slot of 32 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b100	1	2 slots of 16 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	3.072 MHz	6.144 MHz	384 kHz	15	0b100	0	1 slots of 16 bits per frame
		3.072 MHz	6.144 MHz	384 kHz	15	0b010	1	2 slots of 8 bits per frame
16 kHz	up to 8	1.024 MHz	8.192 MHz	128 kHz	63	0b111	1	2 slots of 32 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b100	3	4 slots of 16 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	1.024 MHz	6.144 MHz	128 kHz	47	0b110	1	2 slots of 24 bits per frame
		1.024 MHz	6.144 MHz	128 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	1.024 MHz	4.096 MHz	128 kHz	31	0b111	0	1 slot of 32 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b100	1	2 slots of 16 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	1.024 MHz	2.048 MHz	128 kHz	15	0b100	0	1 slot of 16 bits per frame
		1.024 MHz	2.048 MHz	128 kHz	15	0b010	1	2 slots of 8 bits per frame

1. Refer to *Table 369: TDM settings* for additional information on TDM configuration. The sai_a_ker_ck clock frequency provided to the SAI should be a multiple of the SCK_A frequency, and MCKDIV should be programmed accordingly.
2. The table above gives allowed settings for a decimation ratio of 64.

Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via SAI_PDMDLY register.

The new delays values become effective after two TDM frames.

40.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

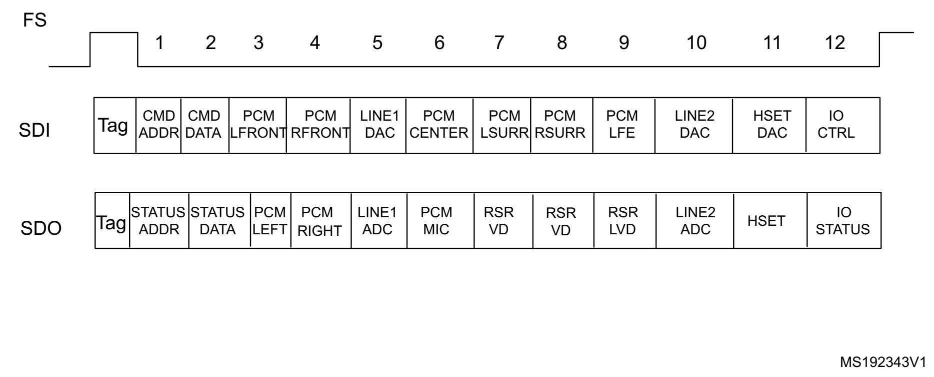
To select this protocol, set PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed to 13 slots. The first one is 16-bit wide and all the others are 20-bit wide (data slots).
- FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

Figure 621 shows an AC’97 audio frame structure.

Figure 621. AC’97 audio frame



Note: In AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

One SAI can be used to target an AC’97 point-to-point communication.

In receiver mode, the SAI acting as an AC’97 link controller requires no FIFO request and so no data storage in the FIFO when the Codec ready bit in the slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC’97 protocol.

Clock generator programming in AC’97 mode

In AC’97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in *Section 40.4.8: SAI clock generator* must be used with FRL = 255, in order to generate the proper frame rate (F_{FS_x}).

40.4.12 SPDIF output

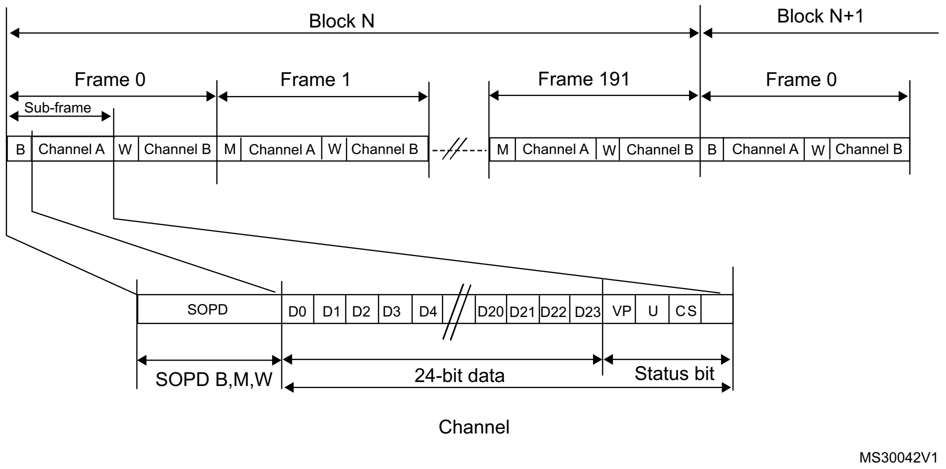
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set PRTCFG[1:0] bit to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only SD data line is enabled.
- FS, SCK, MCLK I/Os pins are left free.
- MODE[1] bit is forced to 0 to select the master mode in order to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock should be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in *Figure 622*.

Figure 622. SPDIF format



A SPDIF block contains 192 frames. Each frame is composed of two 32-bit sub-frames, generally one for the left channel and one for the right channel. Each sub-frame is composed of a SOPD pattern (4-bit) to specify if the sub-frame is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see *Table 371*). The next 28 bits of channel information are composed of 24 bits data + 4 status bits.

Table 371. SOPD pattern

SOPD	Preamble coding		Description
	last bit is 0	last bit is 1	
B	11101000	00010111	Channel A data at the start of block
W	11100100	00011011	Channel B data somewhere in the block
M	11100010	00011101	Channel A data

The data stored in SAI_xDR has to be filled as follows:

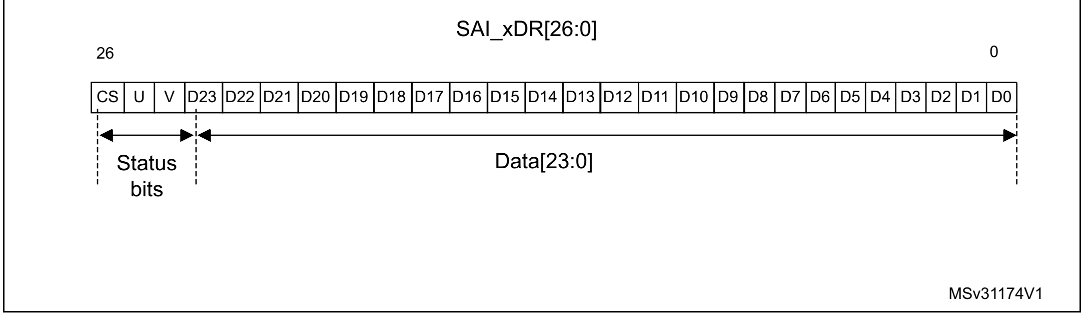
- SAI_xDR[26:24] contain the Channel status, User and Validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, then data must be mapped on SAI_xDR[23:4].

If the data size is 16 bits, then data must be mapped on SAI_xDR[23:8].

SAI_xDR[23] always represents the MSB.

Figure 623. SAI_xDR register ordering



Note: The transfer is performed always with LSB first.

The SAI first sends the adequate preamble for each sub-frame in a block. The SAI_xDR is then sent on the SD line (manchester coded). The SAI ends the sub-frame by transferring the Parity bit calculated as described in *Table 372*.

Table 372. Parity bit calculation

SAI_xDR[26:0]	Parity bit P value transferred
odd number of 0	0
odd number of 1	1

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence should be executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.

   The software needs to point to the address of the future data corresponding to a start of new block (data for preamble B). If the DMA is used, the DMA source base address pointer should be updated accordingly.
5. Enable again the DMA stream (DMA peripheral) if the DMA used to manage data transfers according to the new source base address.
6. Enable again the SAI by setting SAIEN bit in SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice faster as the symbol-rate. The table hereafter shows usual examples of symbol rates with respect to the audio sampling rate.

Table 373. Audio sampling frequency versus symbol rates

Audio sampling frequencies (F_S)	Symbol-rate
44.1 kHz	2.8224 MHz
48 kHz	3.072 MHz
96 kHz	6.144 MHz
192 kHz	12.288 MHz

More generally, the relationship between the audio sampling frequency (F_S) and the bit clock rate (F_{SCK_X}) is given by the formula:

$$F_S = \frac{F_{SCK\_x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

Note: The above formulas are valid only if NODIV is set to 1 in SAI_ACR1 register.

40.4.13 Specific features

The SAI interface embeds specific features which can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at anytime. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is set during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, MUTEVAL bit in the SAI_xCR2 is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed, without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode,

1. Set MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2 as in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

Companding mode

Telecommunication applications can require to process the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when Free protocol mode is selected), the application software can choose to process or not the data before sending it on SD serial output line (compression) or to expand the data after the reception on SD serial input line (expansion) as illustrated in *Figure 624*. The two companding modes supported are the µ-Law and the A-Law log which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

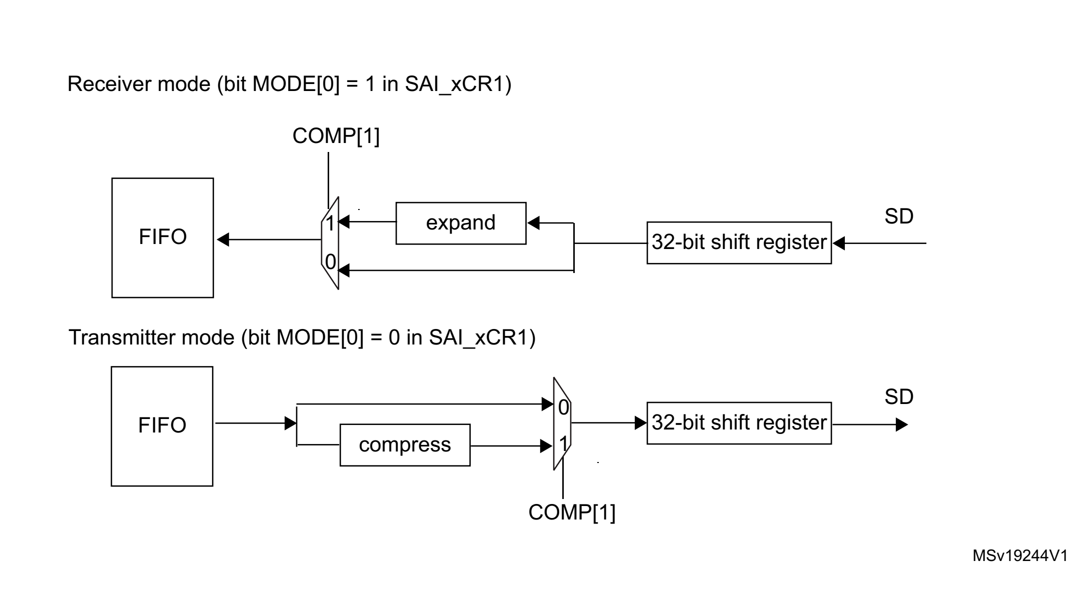
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law or A-Law companding standard can be computed based on 1’s complement or 2’s complement representation depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8-bit wide. For this reason, DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes selected through the COMP[1:0] bits.

If no companding processing is required, COMP[1:0] bits should be kept clear.

Figure 624. Data companding hardware in an audio block in the SAI



1. Not applicable when AC'97 or SPDIF are selected.

Expansion and compression mode are automatically selected through the SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via TRIS bit).

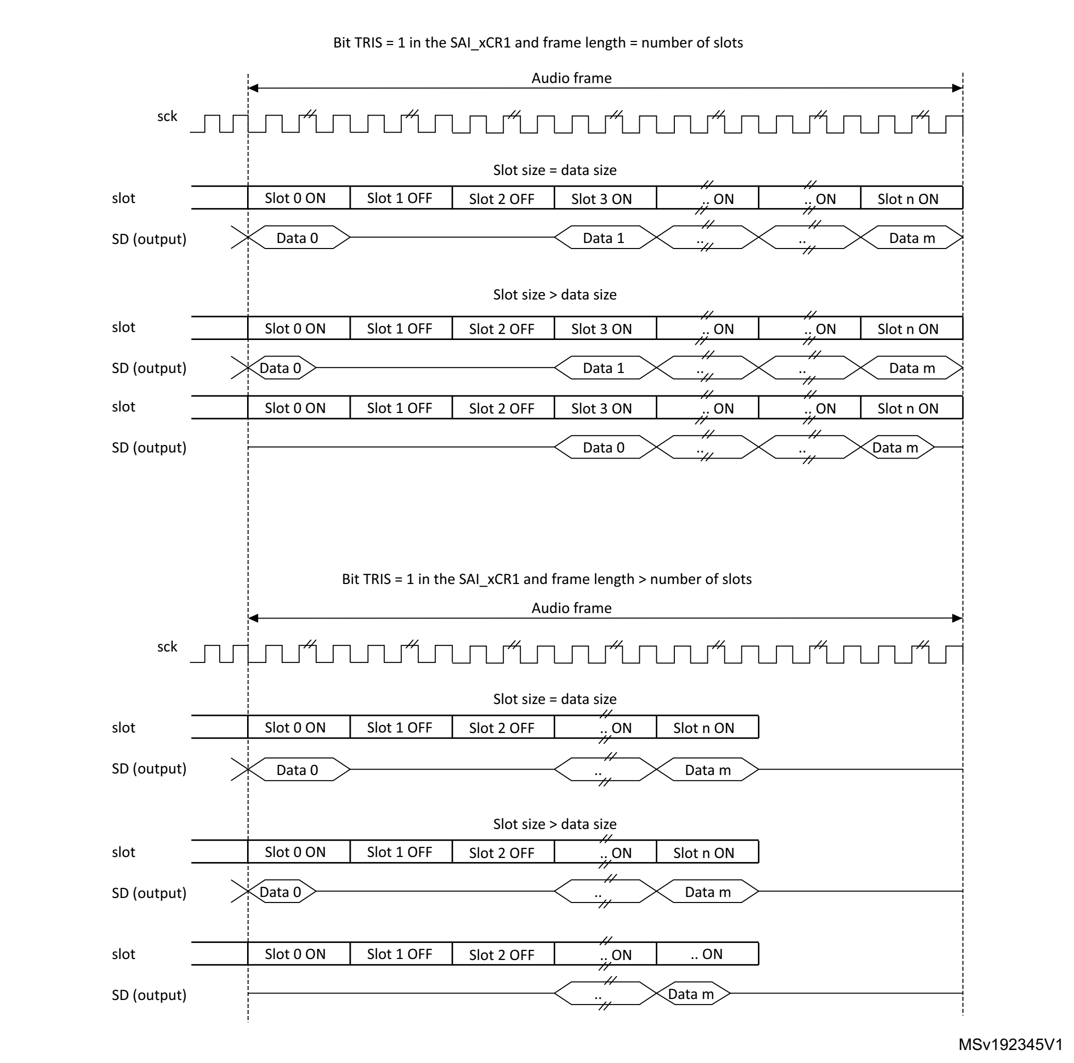
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in HI-z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which could result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame.

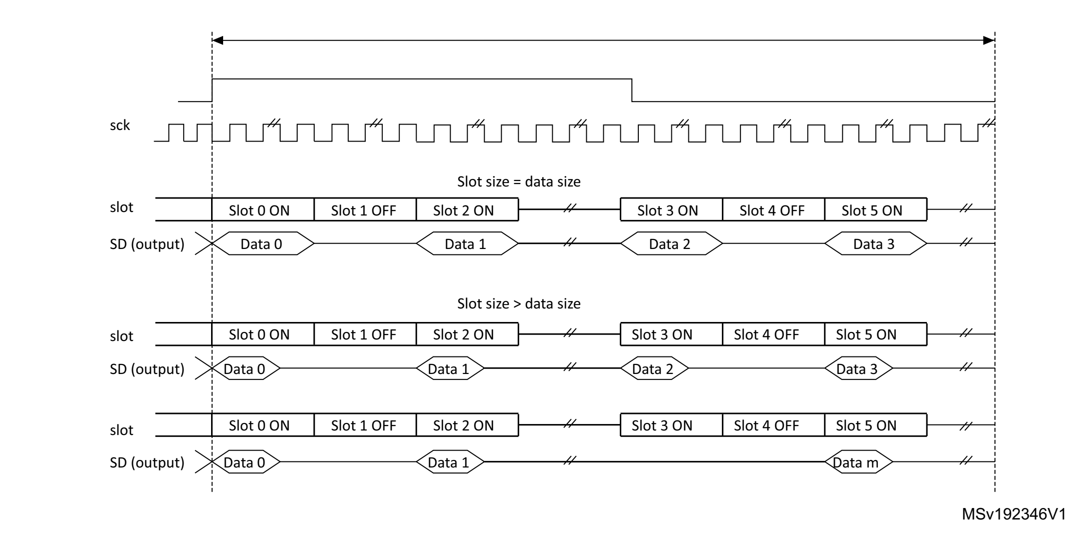
Figure 625 illustrates these behaviors.

Figure 625. Tristate strategy on SD output line on an inactive slot



When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to *Figure 626* (where bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).

Figure 626. Tristate on output data line in a protocol like I2S



If the TRIS bit in the SAI_xCR2 register is cleared, all the High impedance states on the SD output line on *Figure 625* and *Figure 626* are replaced by a drive with a value of 0.

40.4.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun
- Anticipated frame synchronization detection
- Late frame synchronization detection
- Codec not ready (AC’97 exclusively)
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

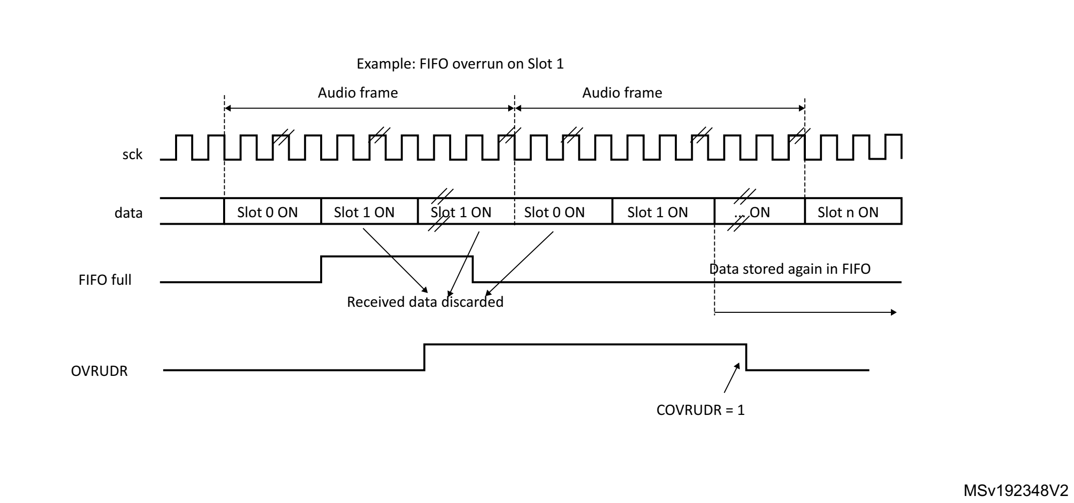
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the flag OVRUDR in the SAI_xSR register is set and an interrupt is generated if OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from new audio frame) from the slot number which was stored internally when the overrun condition was detected. This avoids data slot de-alignment in the destination memory (refer to *Figure 627*).

The OVRUDR flag is cleared when COVRUDR bit is set in the SAI_xCLRFR register.

Figure 627. Overrun detection error



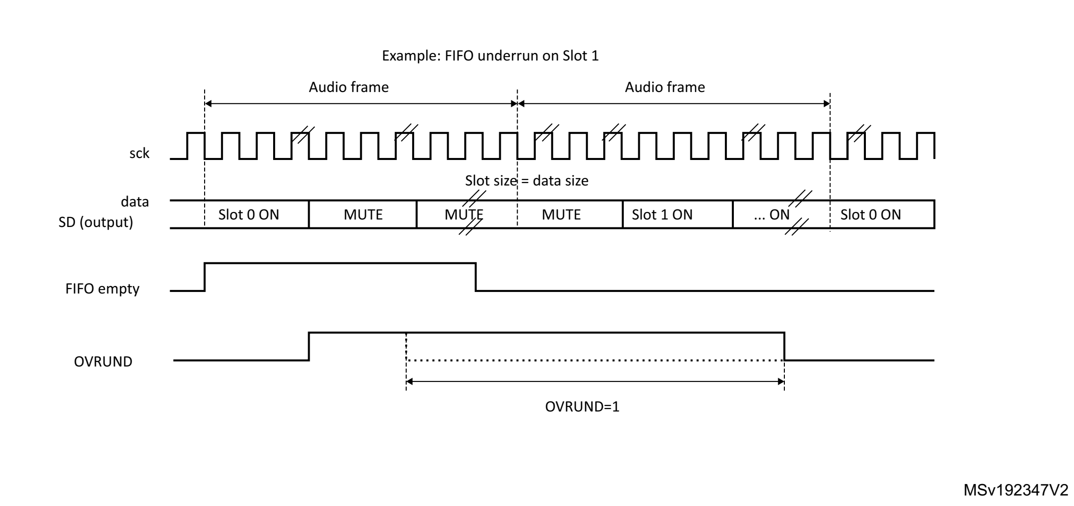
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to *Figure 628*). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 628. FIFO underrun event



Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting SAIEN bit in SAI_xCR1 register. To make sure the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via FFLUS bit in SAI_xCR2 register.
3. Enable again the SAI peripheral (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity and the frame offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expecting time thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in *Anticipated frame synchronization detection (AFSDET)*).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which should not be the case in most applications. In this case, the LFSDET flag is set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the signal FS is not used by the protocol.

Codec not ready (CNRDY AC’97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC’97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the Codec is not ready to communicate during the reception of the TAG 0 (slot0) of the AC’97 audio frame. In this case, no data are automatically stored into the FIFO since the Codec is not ready, until the TAG 0 indicates that the Codec is ready. All the active slots defined in the SAI_xSLOTR register are captured when the Codec is ready.

To clear CNRDY flag, CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

MODE, NODIV, and SAIEN bits belong to SAI_xCR1 register and FRL to SAI_xFRCR register.

If WCKCFGIE bit is set, an interrupt is generated when WCKCFG flag is set in the SAI_xSR register. To clear this flag, set CWCKCFG bit in the SAI_xCLRFR register.

When WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of SAIEN bit.

40.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI is stops working. SAIEN bit remains High until the SAI is completely switched-off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one which is the master must be disabled first.

40.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to *Section 40.4.9: Internal FIFOs*). DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block is operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:

1. Configure SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

Note: Before configuring the SAI block, the SAI DMA channel must be disabled.

40.5 SAI interrupts

The SAI supports 7 interrupt sources as shown in *Table 374*.

Table 374. SAI interrupt sources

Interrupt acronym	Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
SAI	FREQ	FREQ	Master or slave Receiver or transmitter	FREQIE in SAI_xIM register	Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver) For more details refer to *Section 40.4.9: Internal FIFOs*
	OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
	AFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	AFSDETIE in SAI_xIM register	CAFSDET = 1 in SAI_xCLRFR register
	LFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	LFSDETIE in SAI_xIM register	CLFSDET = 1 in SAI_xCLRFR register
	CNRDY	ERROR	Slave (only in AC'97 mode)	CNRDYIE in SAI_xIM register	CCNRDY = 1 in SAI_xCLRFR register
	MUTEDET	MUTE	Master or slave Receiver mode only	MUTEDETIE in SAI_xIM register	CMUTEDET = 1 in SAI_xCLRFR register
	WCKCFG	ERROR	Master with NODIV = 0 in SAI_xCR1 register	WCKCFGIE in SAI_xIM register	CWCKCFG = 1 in SAI_xCLRFR register

Follow the sequence below to enable an interrupt:

1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

40.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

40.6.1 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x004

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				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.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable

0: The master clock is not generated

1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock

This bit is meaningful only when NODIV bit is set to 0.

0: Master clock frequency = F_{FS} x 256

1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider

These bits are set and cleared by software.

000000: Divides by 1 the kernel clock input (sai_x_ker_ck).

Otherwise, The master clock frequency is calculated according to the formula given in *Section 40.4.8: SAI clock generator*.

These bits have no meaning when the audio block is slave.

They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider

This bit is set and cleared by software.

0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512

1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable

This bit is set and cleared by software.

0: DMA disabled

1: DMA enabled

Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable

This bit is set by software.

To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.

This bit allows controlling the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.

0: SAI audio block disabled

1: SAI audio block enabled.

Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive

This bit is set and cleared by software.

0: Audio block output driven when SAIEN is set

1: Audio block output driven immediately after the setting of this bit.

Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: Reserved.

11: Reserved

Note: The audio subblock should be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC’97 audio protocol since AC’97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved
001: Reserved
010: 8 bits
011: 10 bits
100: 16 bits
101: 20 bits
110: 24 bits
111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol allows to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.
01: SPDIF protocol
10: AC’97 protocol
11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter
01: Master receiver
10: Slave transmitter
11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

40.6.2 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x024

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				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.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable

0: The master clock is not generated

1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock

This bit is meaningful only when NODIV bit is set to 0.

0: Master clock frequency = F_{FS} x 256

1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider

These bits are set and cleared by software.

000000: Divides by 1 the kernel clock input (sai_x_ker_ck).

Otherwise, The master clock frequency is calculated according to the formula given in *Section 40.4.8: SAI clock generator*.

These bits have no meaning when the audio block is slave.

They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider

This bit is set and cleared by software.

0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512

1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable

This bit is set and cleared by software.

0: DMA disabled

1: DMA enabled

Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable

This bit is set by software.

To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.

This bit allows controlling the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.

0: SAI audio block disabled

1: SAI audio block enabled.

Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive

This bit is set and cleared by software.

0: Audio block output driven when SAIEN is set

1: Audio block output driven immediately after the setting of this bit.

Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: Reserved.

11: Reserved

Note: The audio subblock should be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC’97 audio protocol since AC’97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol allows to use the powerful config uration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC’97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.

40.6.3 SAI configuration register 2 (SAI_ACR2)

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
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software.It must be written before enabling the audio block: SAIEN.

This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It should be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit should be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

40.6.4 SAI configuration register 2 (SAI_BCR2)

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
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software.It must be written before enabling the audio block: SAIEN.

This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It should be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit should be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

40.6.5 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x00C

Reset value: 0x0000 0007

Note: This register has no meaning in AC’97 and SPDIF audio protocol.

31	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.	FSOFF	FSPOL	FSDEF
													rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC’97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.

0: FS is asserted on the first bit of the slot 0.

1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.

This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC’97 or SPDIF audio block configuration.

This bit must be configured when the audio block is disabled.

0: FS is active low (falling edge)

1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.

This bit is set and cleared by software.

0: FS signal is a start frame signal

1: FS signal is a start of frame signal + channel side identification

When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).

This bit is meaningless and is not used in AC’97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length should be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

40.6.6 SAI frame configuration register (SAI_BFRCR)

Address offset: 0x02C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	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.	FSOFF	FSPOL	FSDEF
													rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.

0: FS is asserted on the first bit of the slot 0.

1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.

This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.

This bit must be configured when the audio block is disabled.

0: FS is active low (falling edge)

1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.

This bit is set and cleared by software.

0: FS signal is a start frame signal

1: FS signal is a start of frame signal + channel side identification

When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).

This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length should be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

40.6.7 SAI slot register (SAI_ASLOTR)

Address offset: 0x010

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[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.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

0: Inactive slot.

1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots should be even if FSDEF bit in the SAI_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 7:6 **SLOTSZ[1:0]**: Slot size

This bits is set and cleared by software.

The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.

Refer to *Output data line management on an inactive slot* for information on how to drive SD line.

These bits must be set when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).

01: 16-bit

10: 32-bit

11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset

These bits are set and cleared by software.

The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.

These bits must be set when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

40.6.8 SAI slot register (SAI_BSLOTR)

Address offset: 0x030

Reset value: 0x0000 0000

Note: This register has no meaning in AC’97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[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.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.
The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots should be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.

Bits 7:6 **SLOTSZ[1:0]**: Slot size

This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset

These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.

40.6.9 SAI interrupt mask register (SAI_AIM)

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.	Res.	LFSDET IE	AFSDETI E	CNRDY IE	FREQ IE	WCKCFG IE	MUTEDET IE	OVRUDR IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.

Note: This bit is used only in Free protocol mode *and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

40.6.10 SAI interrupt mask register (SAI_BIM)

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**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET IE	AFSDETI E	CNRDY IE	FREQ IE	WCKCFG IE	MUTEDET IE	OVRUDR IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.

This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.

This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC’97).

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.

This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.

Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode, Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.

Note: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

40.6.11 SAI status register (SAI_ASR)

Address offset: 0x018

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.	FLVL[2:0]		
													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.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

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

Bit 6 **LFSDET**: Late frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.

This bit is read only.
0: External AC’97 Codec is ready
1: External AC’97 Codec is not ready
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.

This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:

– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.

This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.

This bit is read only.

0: Clock configuration is correct

1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 40.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)

This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.

It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.

This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.

This bit is read only.

0: No MUTE detection on the SD input line

1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame

This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).

It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.

This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.

This bit is read only.

0: No overrun/underrun error.

1: Overrun/underrun error detection.

The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.

It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.

This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

40.6.12 SAI status register (SAI_BSR)

Address offset: 0x038

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.	FLVL[2:0]		
													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.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

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

Bit 6 **LFSDET**: Late frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.

This bit is read only.
0: External AC’97 Codec is ready
1: External AC’97 Codec is not ready
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.

This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:

– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.

This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.

This bit is read only.

0: Clock configuration is correct

1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 40.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)

This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.

It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.

This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.

This bit is read only.

0: No MUTE detection on the SD input line

1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame

This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).

It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.

This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.

This bit is read only.

0: No overrun/underrun error.

1: Overrun/underrun error detection.

The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.

It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.

This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

40.6.13 SAI clear flag register (SAI_ACLRFR)

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.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.

This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.

This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.

This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.

This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.

This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.

This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

40.6.14 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x03C

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.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.

This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC'97or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.

This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC'97or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.

This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.

This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.

This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.

This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

40.6.15 SAI data register (SAI_ADR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[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
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

40.6.16 SAI data register (SAI_BDR)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[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
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

40.6.17 SAI PDM control register (SAI_PDMCR)

Address offset: 0x0044

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.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
				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.

Bit 11 **CKEN4**: Clock enable of bitstream clock number 4

This bit is set and cleared by software.
0: SAI_CK4 clock disabled
1: SAI_CK4 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK4 might not be available for all SAI instances. Refer to Section 40.3: SAI implementation for details.

Bit 10 **CKEN3**: Clock enable of bitstream clock number 3

This bit is set and cleared by software.

0: SAI_CK3 clock disabled

1: SAI_CK3 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK3 might not be available for all SAI instances. Refer to Section 40.3: SAI implementation for details.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2

This bit is set and cleared by software.

0: SAI_CK2 clock disabled

1: SAI_CK2 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK2 might not be available for all SAI instances. Refer to Section 40.3: SAI implementation for details.

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1

This bit is set and cleared by software.

0: SAI_CK1 clock disabled

1: SAI_CK1 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK1 might not be available for all SAI instances. Refer to Section 40.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones

This bit is set and cleared by software.

00: Configuration with 2 microphones

01: Configuration with 4 microphones

10: Configuration with 6 microphones

11: Configuration with 8 microphones

*Note: It is not recommended to configure this field when PDMEN = 1.**

The complete set of data lines might not be available for all SAI instances. Refer to Section 40.3: SAI implementation for details.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable

This bit is set and cleared by software. This bit allows to control the state of the PDM interface block. Make sure that the SAI in already operating in TDM master mode before enabling the PDM interface.

0: PDM interface disabled

1: PDM interface enabled

40.6.18 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]		
	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.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**

This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4

This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 of T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3

This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3

This bit is set and cleared by software.

000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2

This bit is set and cleared by software.

000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2

This bit is set and cleared by software.

000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1

This bit is set and cleared by software.

000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1

This bit is set and cleared by software.

000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.0

40.6.19 SAI register map

The following table summarizes the SAI registers.

Table 375. SAI 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	Reserved																																
0x0004 or 0x0024	SAI_xCR1	Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN	Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
	Reset value					0	0	0	0	0	0	0	0	0		0	0			0	0	0	0	0	0	0	1	0		0	0	0	0
0x0008 or 0x0028	SAI_xCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP[1:0]		CPL	MUTECN[5:0]						MUTE VAL	MUTE	TRIS	FFLUS	FTH[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C or 0x002C	SAI_xFRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF	Res.	FSALL[6:0]							FRL[7:0]							
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0010 or 0x0030	SAI_xSLOTR	SLOTEN[15:0]																Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
	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

Table 375. SAI 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
0x0014 or 0x0034	SAI_xIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
	Reset value																										0	0	0	0	0	0	0
0x0018 or 0x0038	SAI_xSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
	Reset value														0	0	0										0	0	0	0	1	0	0
0x001C or 0x003C	SAI_xCLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
	Reset value																										0	0	0		0	0	0
0x0020 or 0x0040	SAI_xDR	DATA[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
0x0044	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																					0	0	0	0			0	0				0
0x0048	SAI_PDMDLY	Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]			Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[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

Refer to *Section 2.3 on page 129* for the register boundary addresses.

41 Inter-integrated circuit (I2C) interface

41.1 Introduction

The I^{2}C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I^{2}C bus. It provides multimaster capability, and controls all I^{2}C 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.

41.2 I2C main features

- I^{2}C 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 41.3: 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.

41.3 I2C implementation

This manual describes the full set of features implemented in I2C peripheral. In the STM32G4 Series devices I2C1, I2C2, I2C3 and I2C4 implement the full set of features as shown in the following table.

Table 376. I2C implementation

I2C features[1]	I2C1	I2C2	I2C3	I2C4
7-bit addressing mode	X	X	X	X
10-bit addressing mode	X	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X	X
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X
Independent clock	X	X	X	X
Wakeup from Stop 1 mode	X	X	X	X
SMBus/PMBus	X	X	X	X

1. X = supported.

41.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it 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.

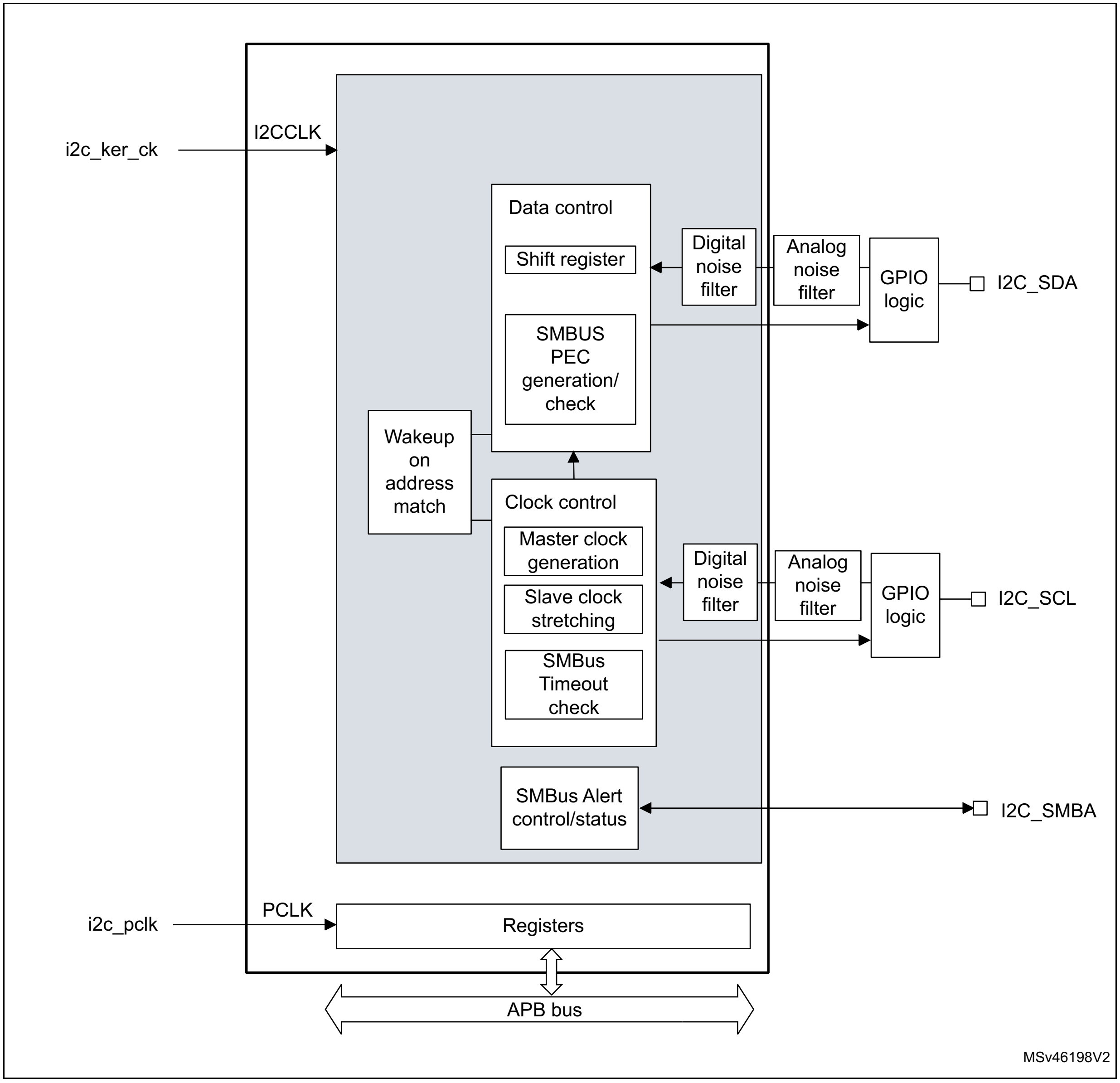
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

41.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 629*.

Figure 629. 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 20mA 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 41.3: I2C implementation.*

41.4.2 I2C pins and internal signals

Table 377. 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 378. 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 392: I2C Interrupt requests* 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)

41.4.3 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}.

41.4.4 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

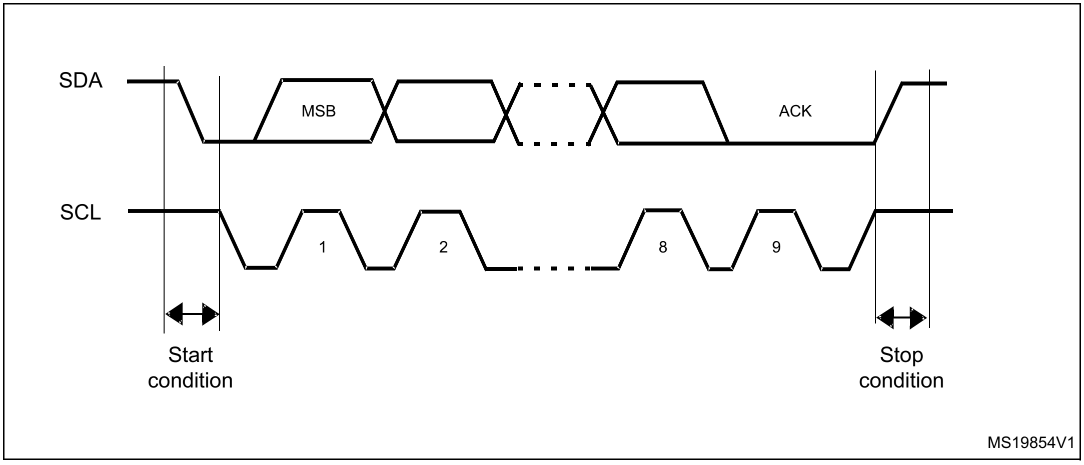
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

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 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Figure 630. I^{2}C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

41.4.5 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^{2}C performs a software reset. Refer to *Section 41.4.6: Software reset* 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^{2}C 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 379. 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

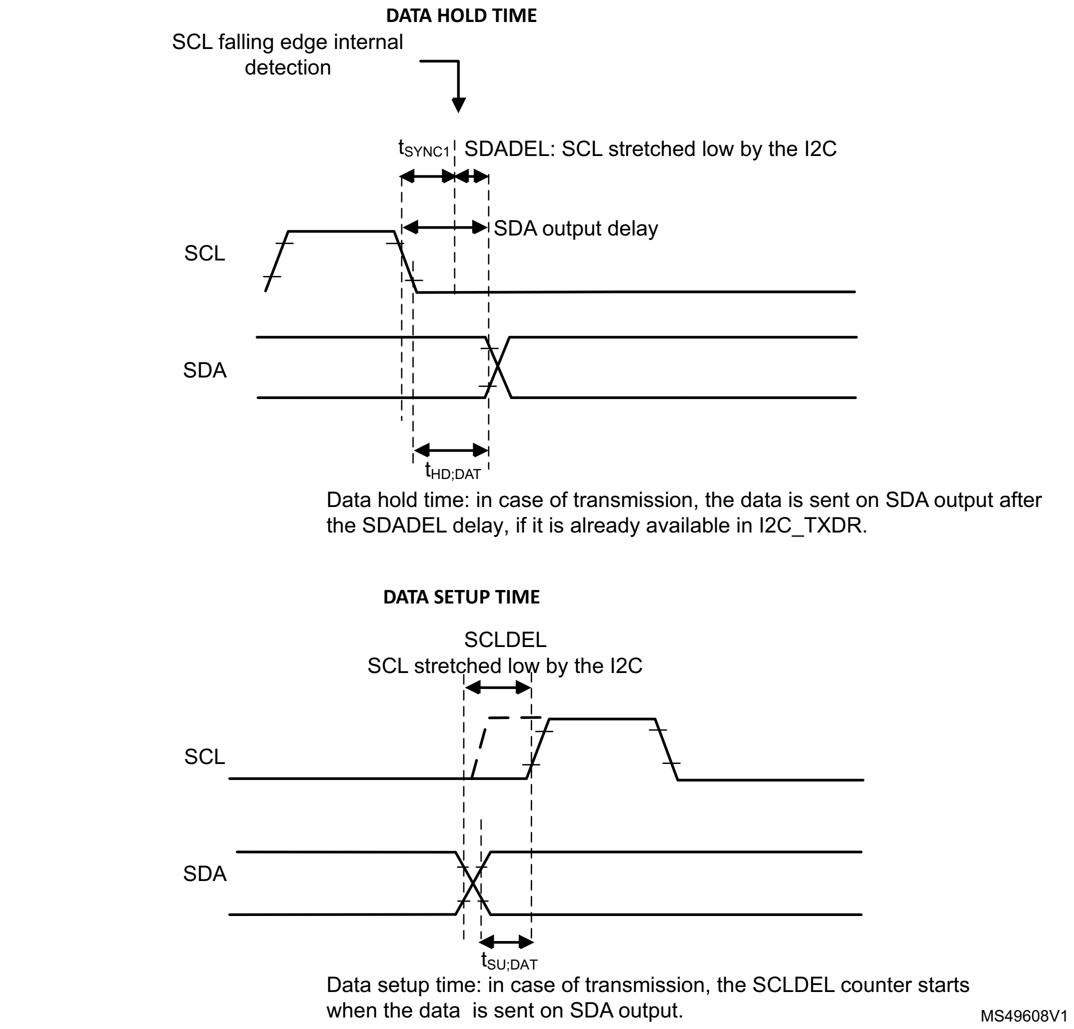
Caution: Changing the filter configuration is not allowed when the I2C is enabled.

I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, 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 631. Setup and hold timings


DATA HOLD TIME
SCL falling edge internal detection
tSYNC1
SDADEL: SCL stretched low by the I2C
SDA output delay
SCL
SDA
tHD;DAT
Data hold time: in case of transmission, the data is sent on SDA output after the SDADEL delay, if it is already available in I2C_TXDR.
DATA SETUP TIME
SCLDEL
SCL stretched low by the I2C
SCL
SDA
tSU;DAT
Data setup time: in case of transmission, the SCLDEL counter starts when the data is sent on SDA output.
MS49608V1

- 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) x t_{I2CCLK}]} / {(PRESC +1) x t_{I2CCLK} } ≤ SDADEL

SDADEL ≤ {$t_{HD;DAT\ (max)}$ -$t_{AF(max)}$ - [(*DNF+4) x t_{I2CCLK}*]} / {(PRESC +1) x 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 ≤ {$t_{VD;DAT\ (max)}$ -$t_{r\ (max)}$ -*260 ns* - [(*DNF+4) x t_{I2CCLK}*]} / {(PRESC +1) x 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 380: I2C-SMBus specification data setup and hold times* 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)] x t_{I2CCLK}]} - 1 <= SCLDEL

Refer to *Table 380: I2C-SMBus specification data setup and hold times* 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, in case 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 380. 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) x t_{PRESC} where t_{PRESC} = (PRESC+1) x 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) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}. t_{SCLH} impacts the SCL high time t_{HIGH} .

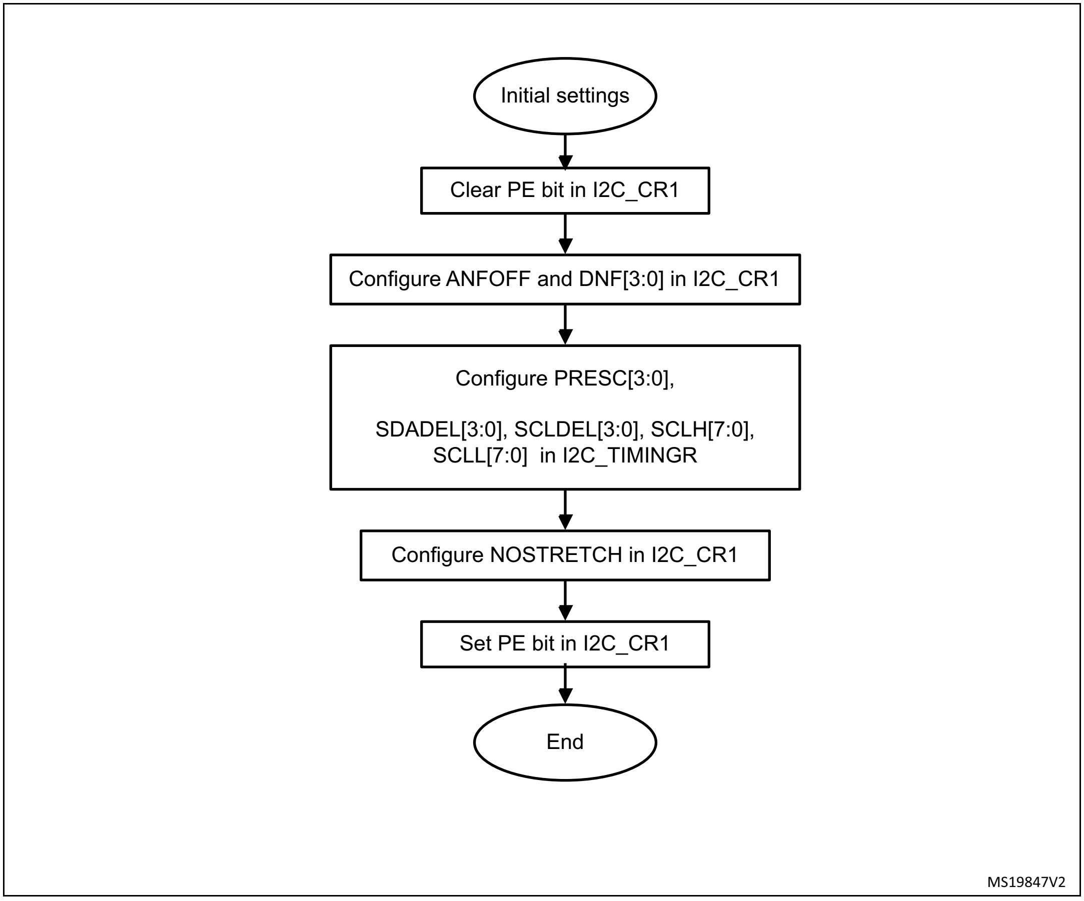
Refer to *I2C master initialization* for more details.

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 632. I2C initialization flowchart


Initial settings
Clear PE bit in I2C_CR1
Configure ANFOFF and DNF[3:0] in I2C_CR1
Configure PRESC[3:0],
SDADEL[3:0], SCLDEL[3:0], SCLH[7:0], SCLL[7:0] in I2C_TIMINGR
Configure NOSTRETCH in I2C_CR1
Set PE bit in I2C_CR1
End
MS19847V2

41.4.6 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 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1.

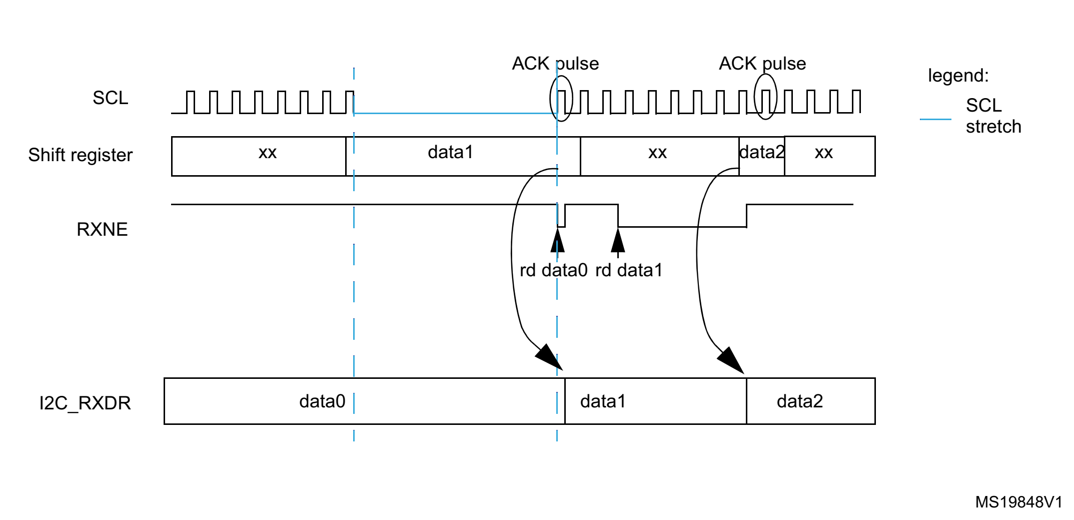
41.4.7 Data transfer

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 8th 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 8th and 9th SCL pulse (before the acknowledge pulse).

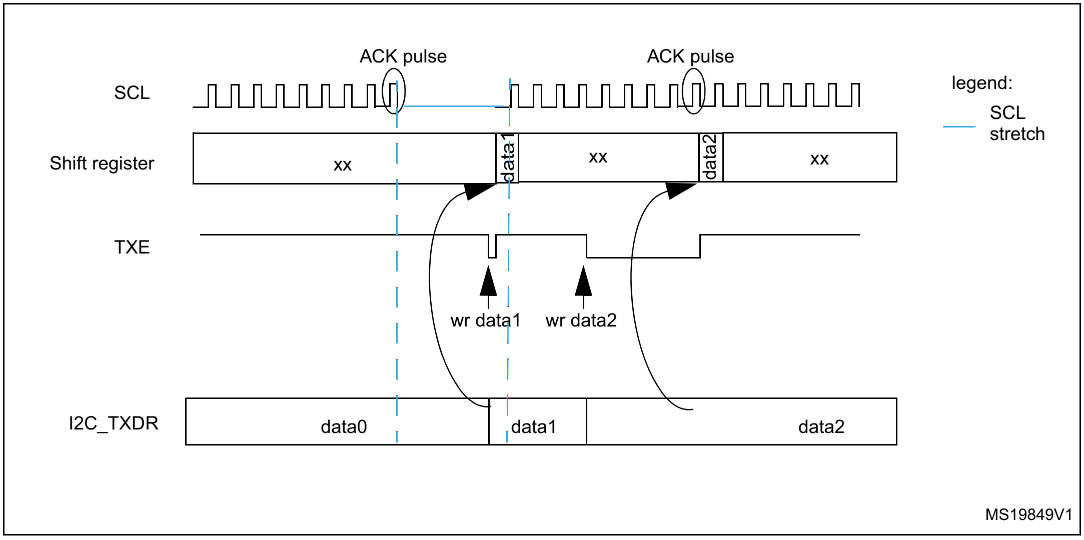
Figure 633. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th 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 9th SCL pulse.

Figure 634. 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_CR2 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 2 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 381. 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

41.4.8 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 2nd 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 9th 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 8th and 9th 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.

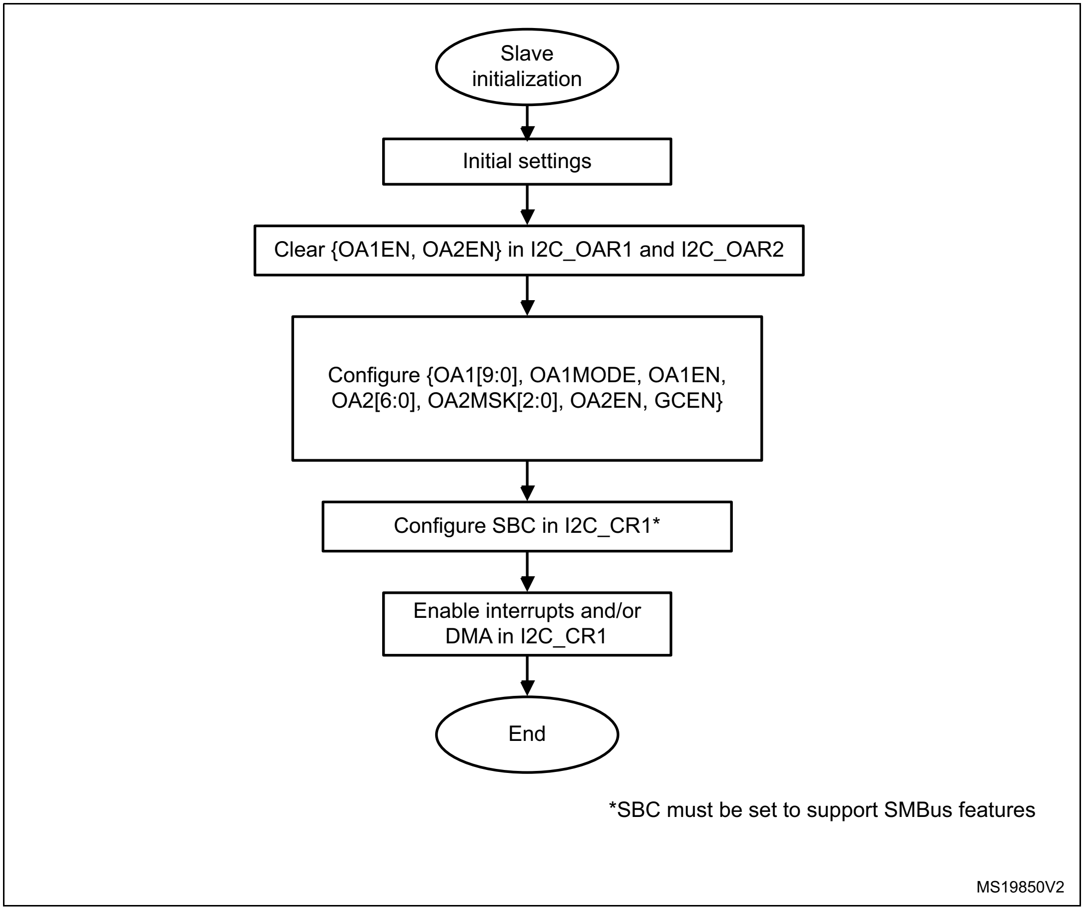
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

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 635. Slave initialization flowchart


Slave initialization
Initial settings
Clear {OA1EN, OA2EN} in I2C_OAR1 and I2C_OAR2
Configure {OA1[9:0], OA1MODE, OA1EN, OA2[6:0], OA2MSK[2:0], OA2EN, GCEN}
Configure SBC in I2C_CR1*
Enable interrupts and/or DMA in I2C_CR1
End
*SBC must be set to support SMBus features
MS19850V2

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 636. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 0

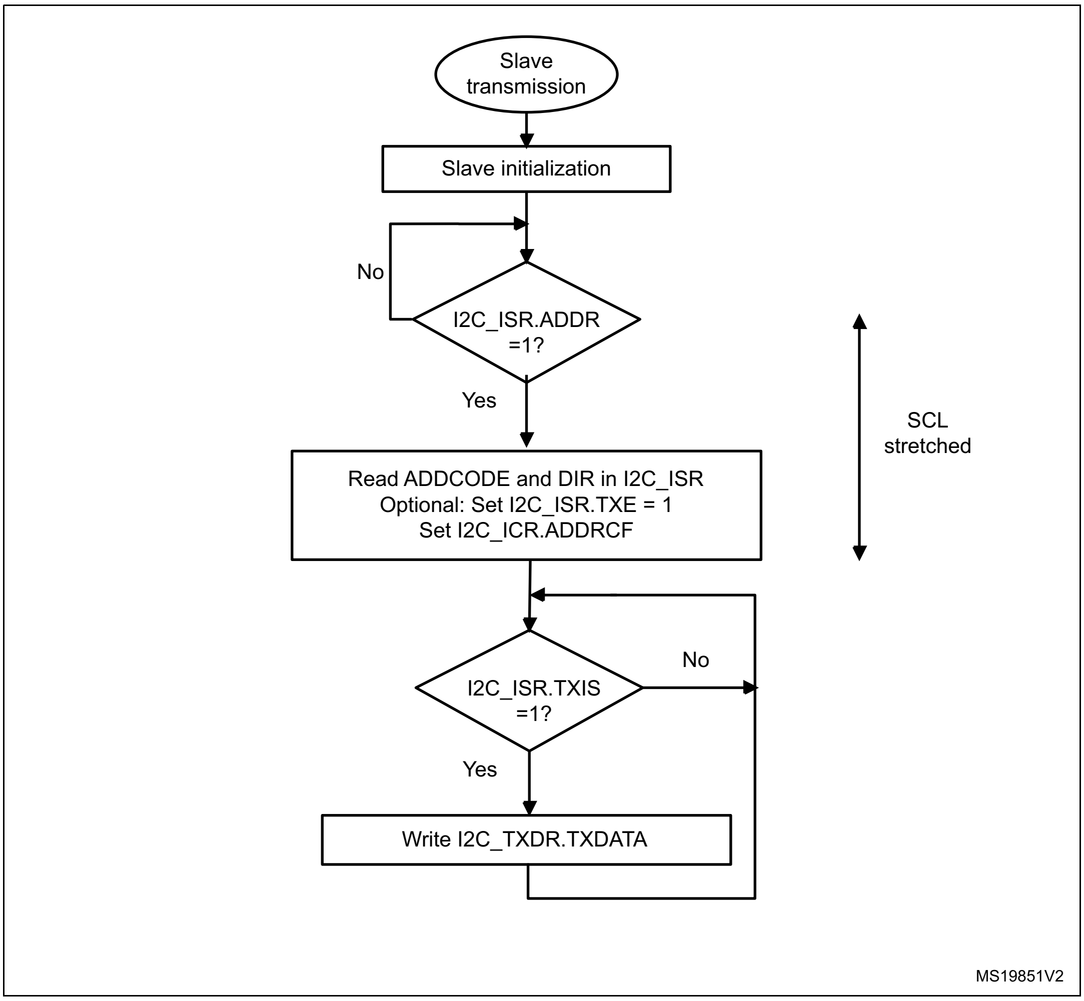


Figure 637. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 1

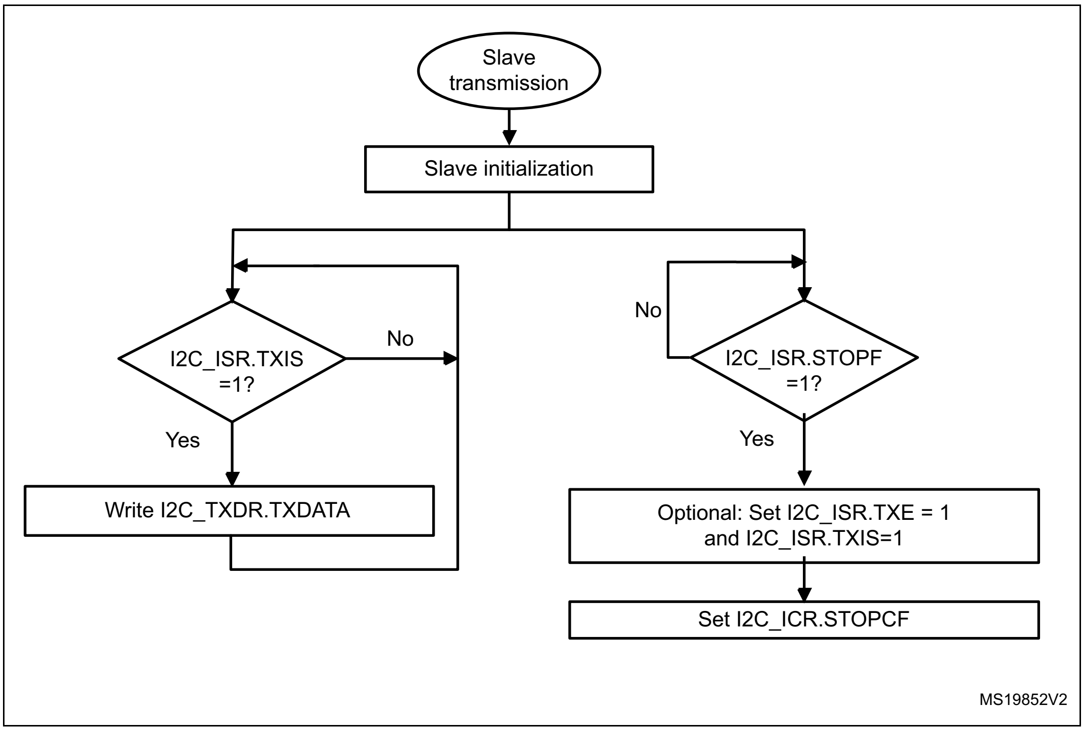
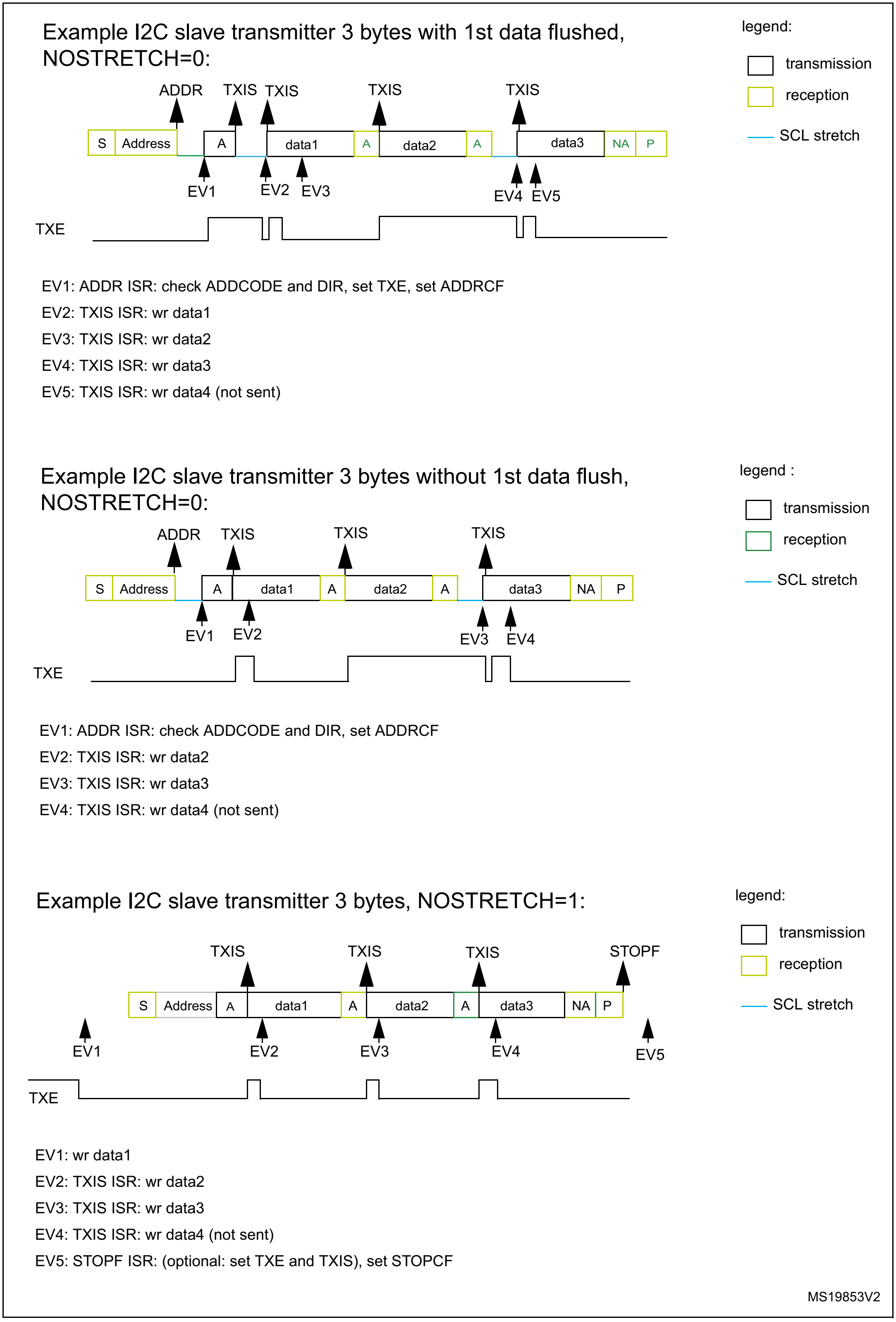


Figure 638. Transfer bus diagrams for I2C slave transmitter



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

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 639. Transfer sequence flowchart for slave receiver with NOSTRETCH=0

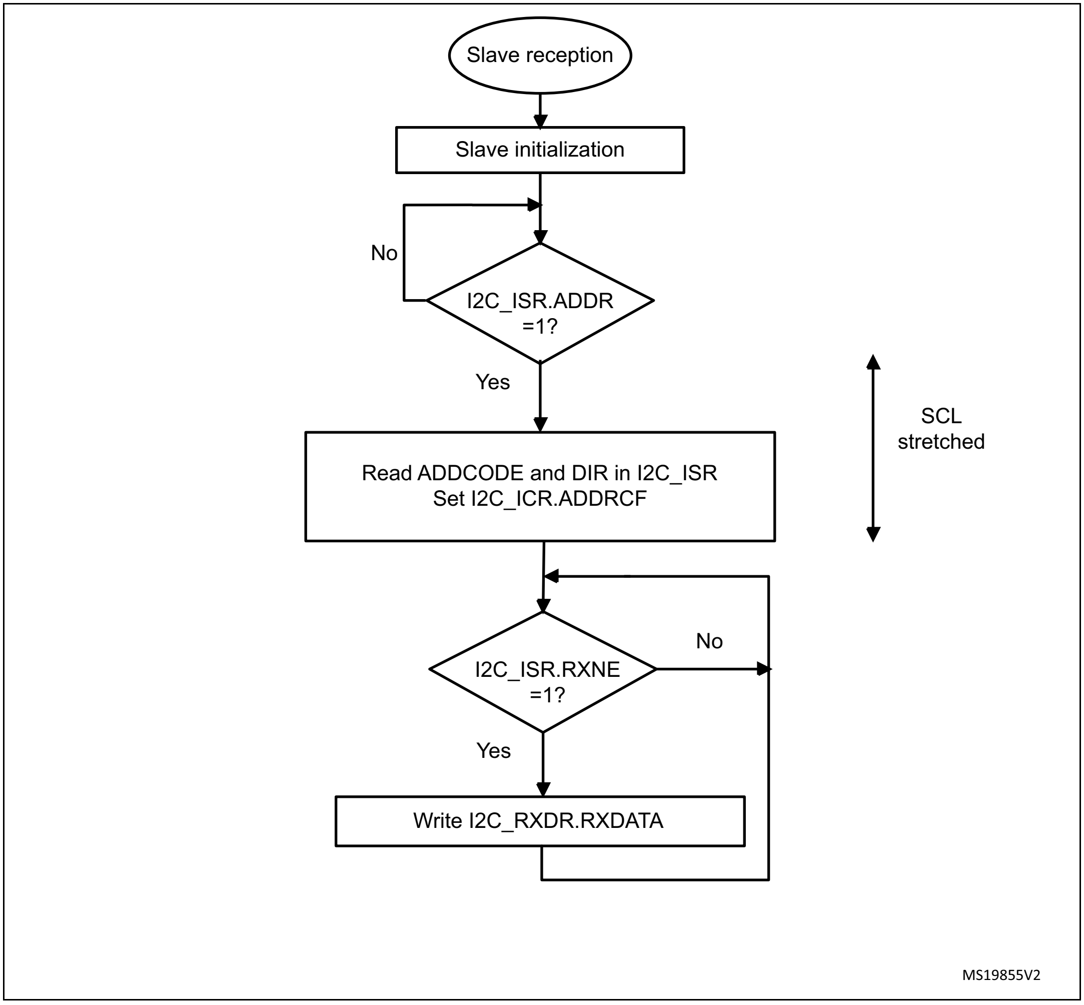


Figure 640. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

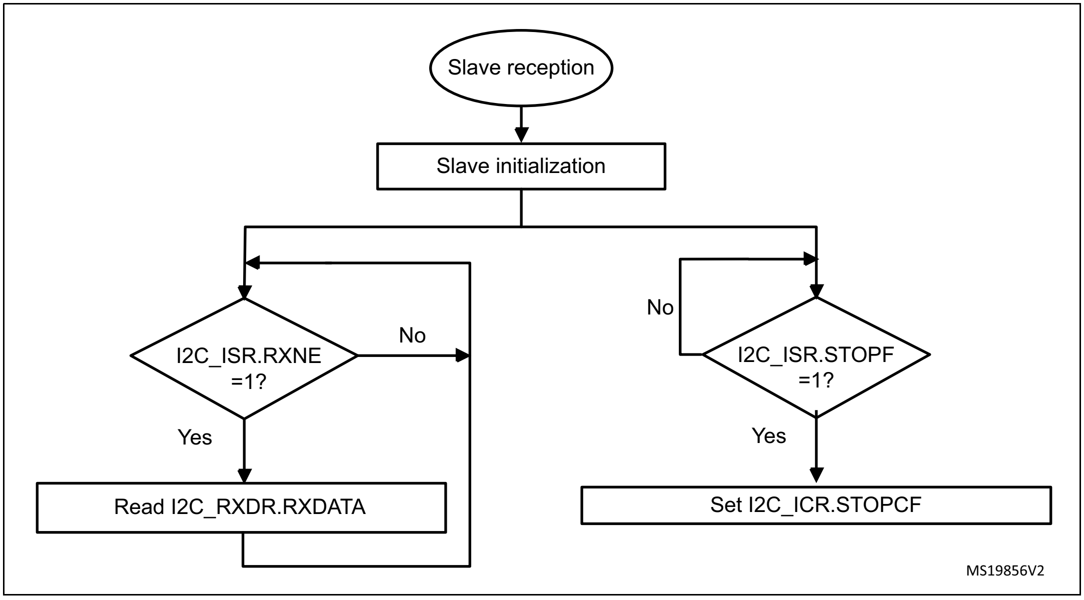
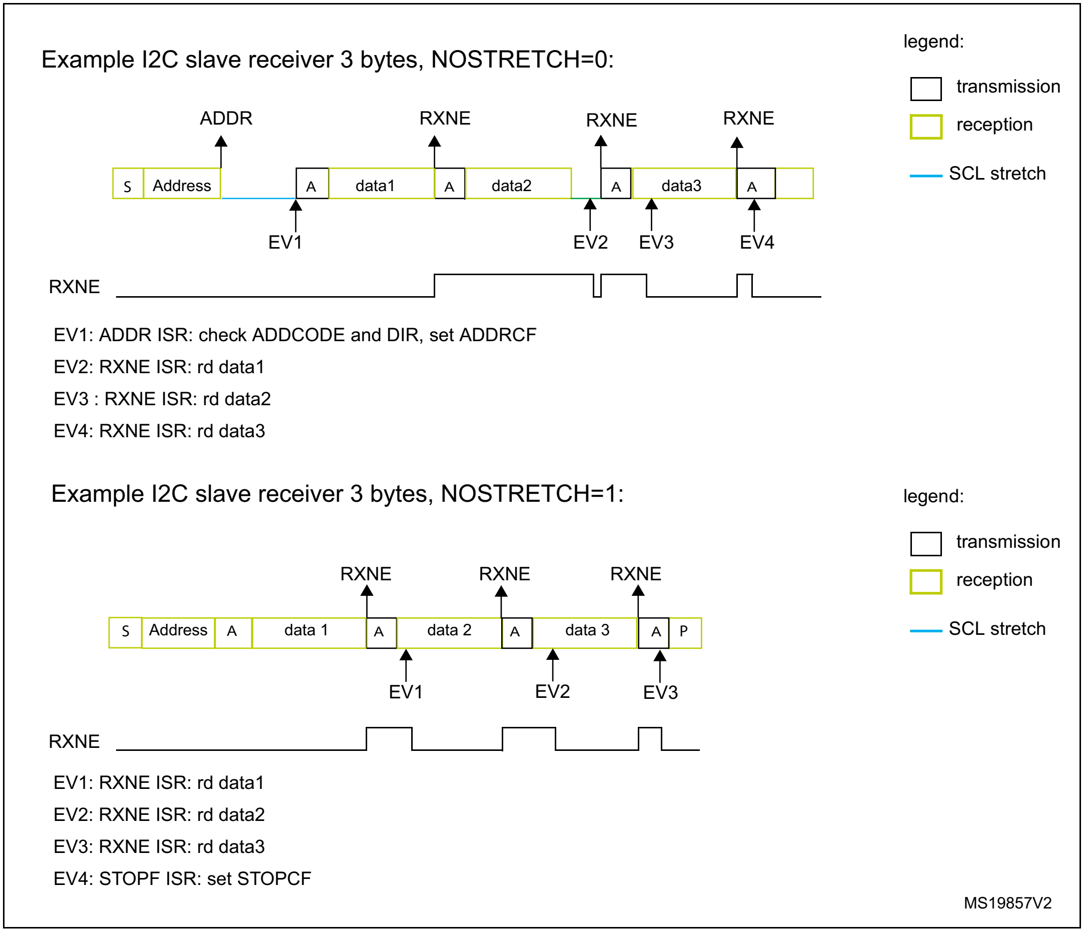


Figure 641. Transfer bus diagrams for I2C slave receiver



EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF

EV2: RXNE ISR: rd data1

EV3 : RXNE ISR: rd data2

EV4: RXNE ISR: rd data3

EV1: RXNE ISR: rd data1

EV2: RXNE ISR: rd data2

EV3: RXNE ISR: rd data3

EV4: STOPF ISR: set STOPCF

41.4.9 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:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$

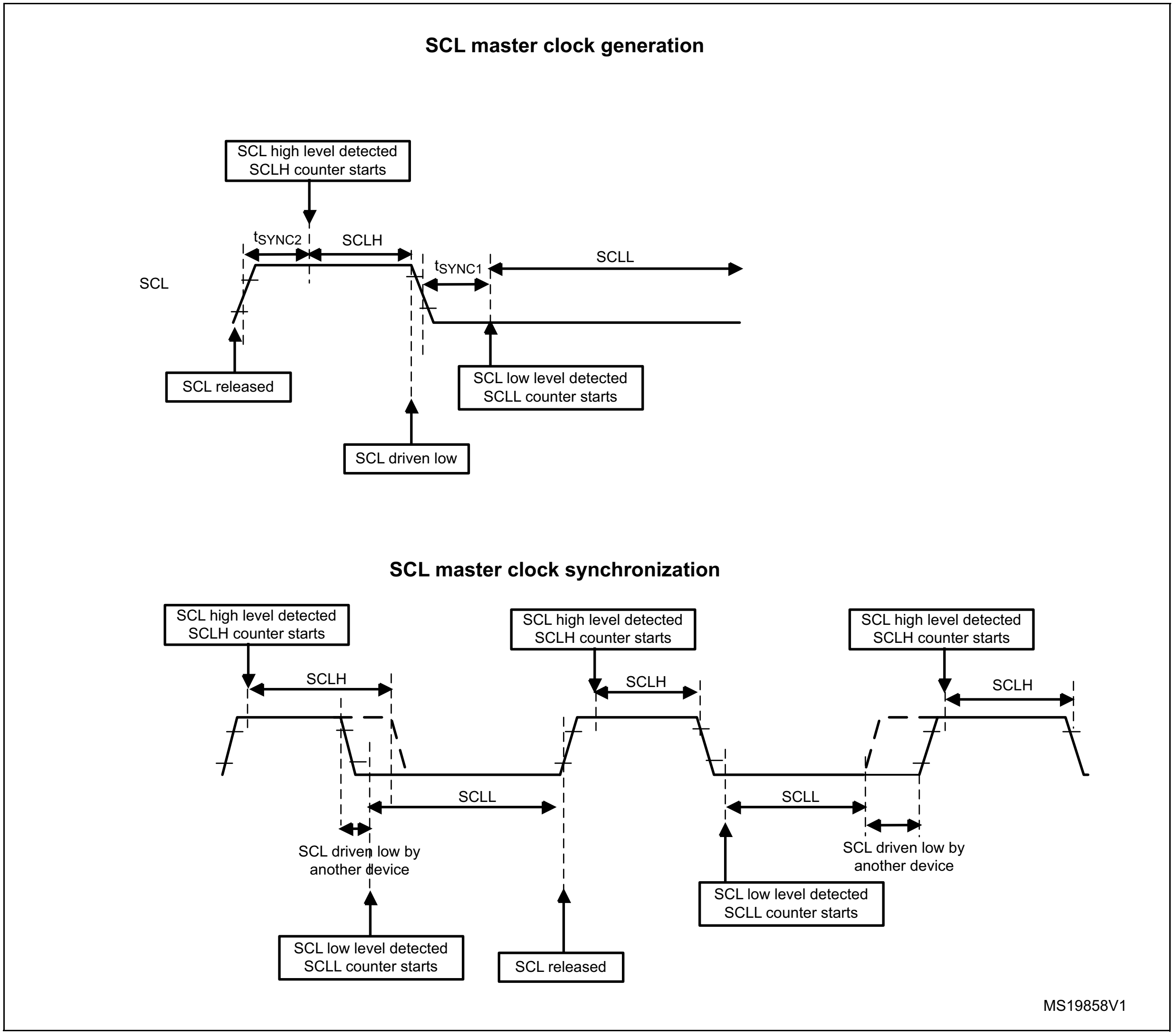
The duration of t_{SYNC1} depends on these parameters:

- 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 642. Master clock generation



Caution: In order to be I^{2}C or SMBus compliant, the master clock must respect the timings given the table below.

Table 382. I^{2}C-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	-	µs
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	µs
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	µs
t_{HIGH}	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	µs
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	ns

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 41.4.10: 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}.

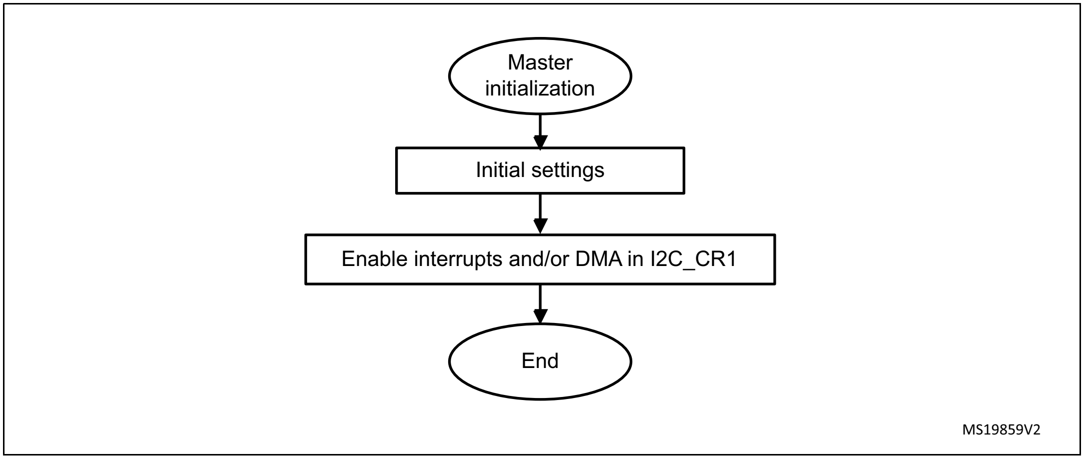
In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

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 is 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, when the ADDRCF bit is set.

Note: The same procedure is applied for a Repeated Start condition. In this case BUSY=1.

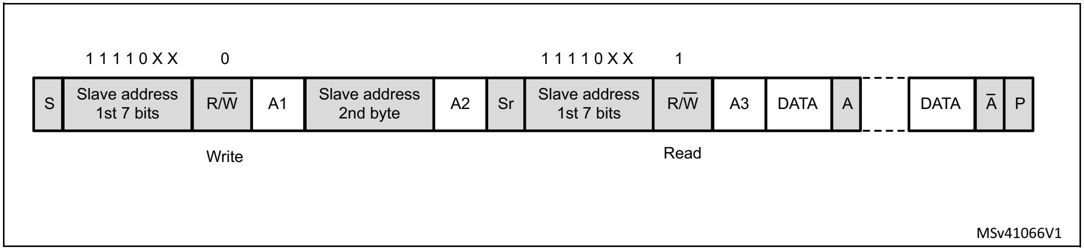
Figure 643. Master initialization flowchart



Initialization of a master receiver addressing a 10-bit address slave

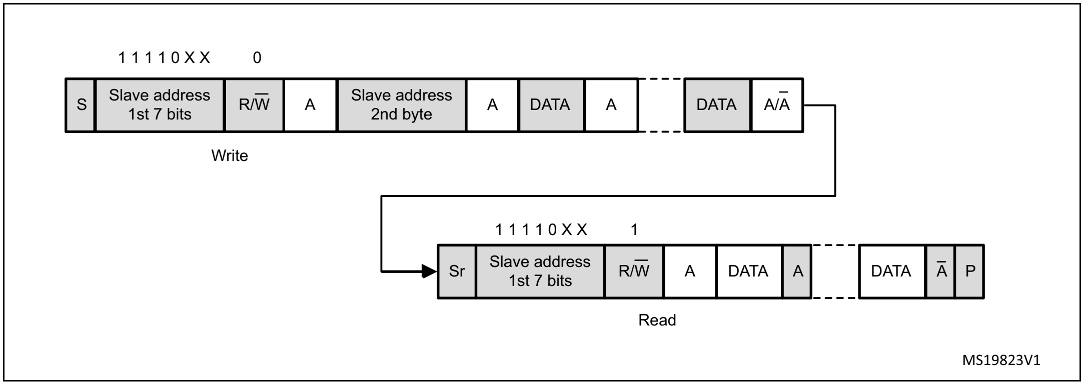
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address 2nd byte + REStart + Slave address 10-bit header Read

Figure 644. 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 645. 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 9th 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 646. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes

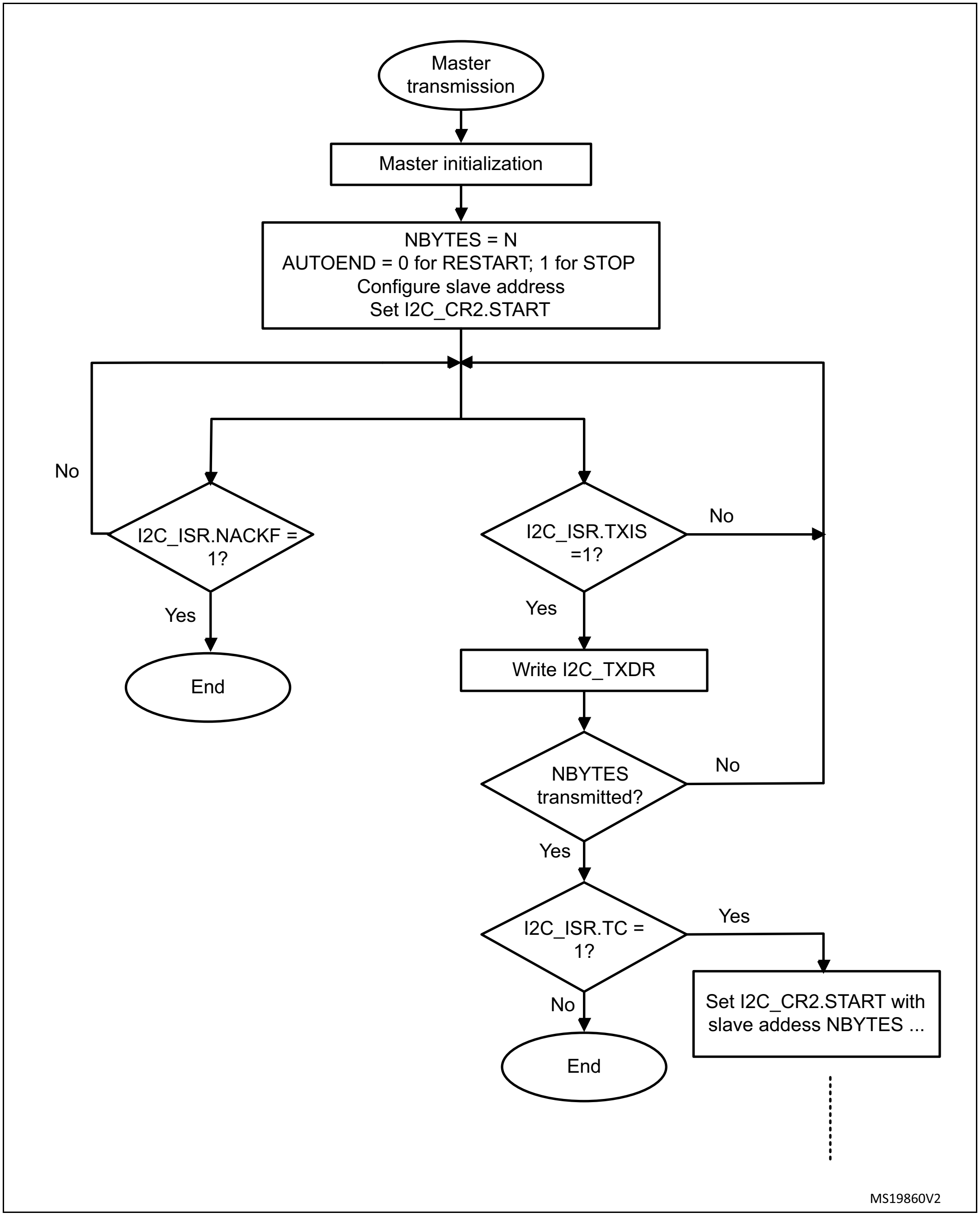


Figure 647. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

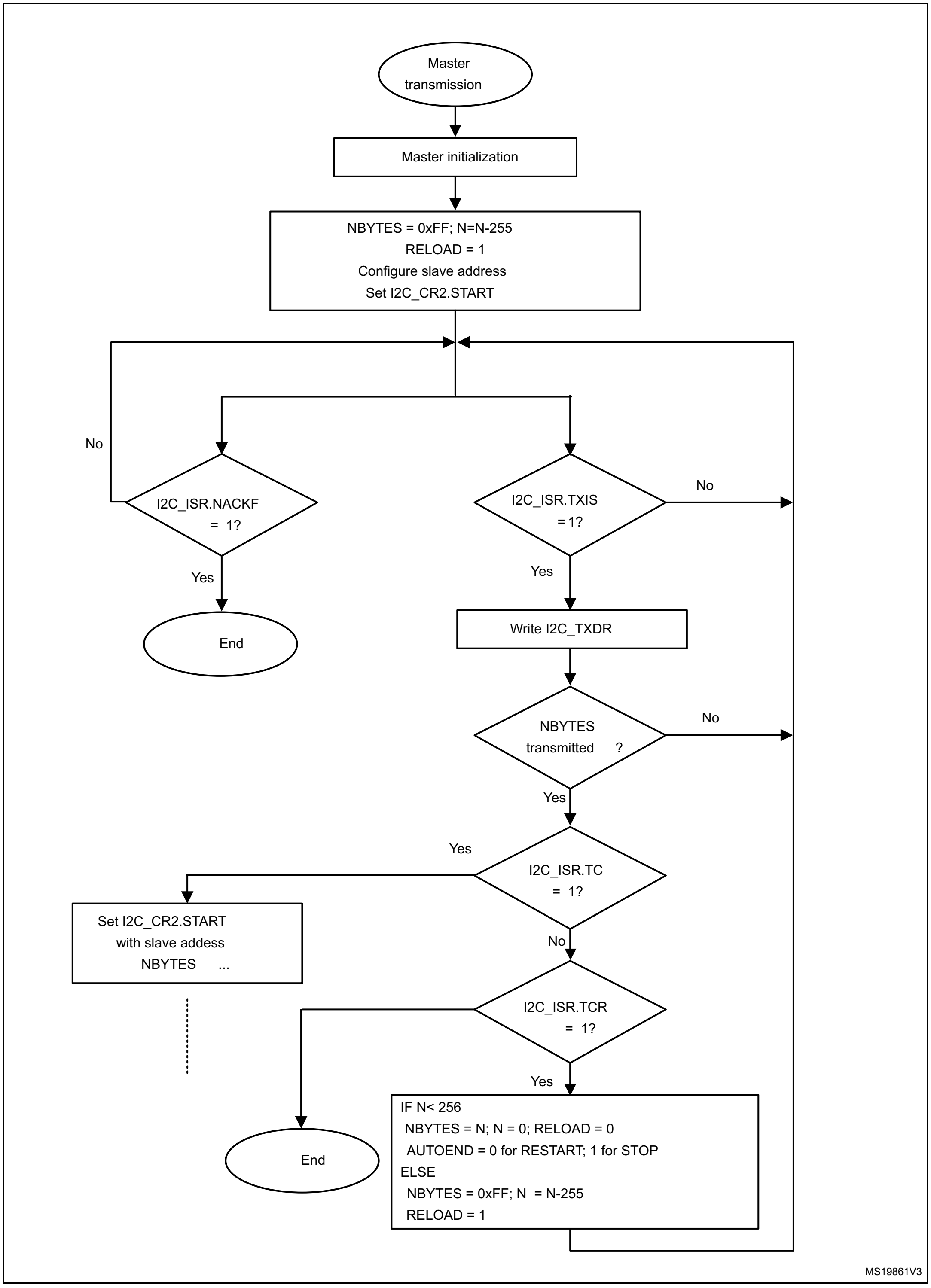
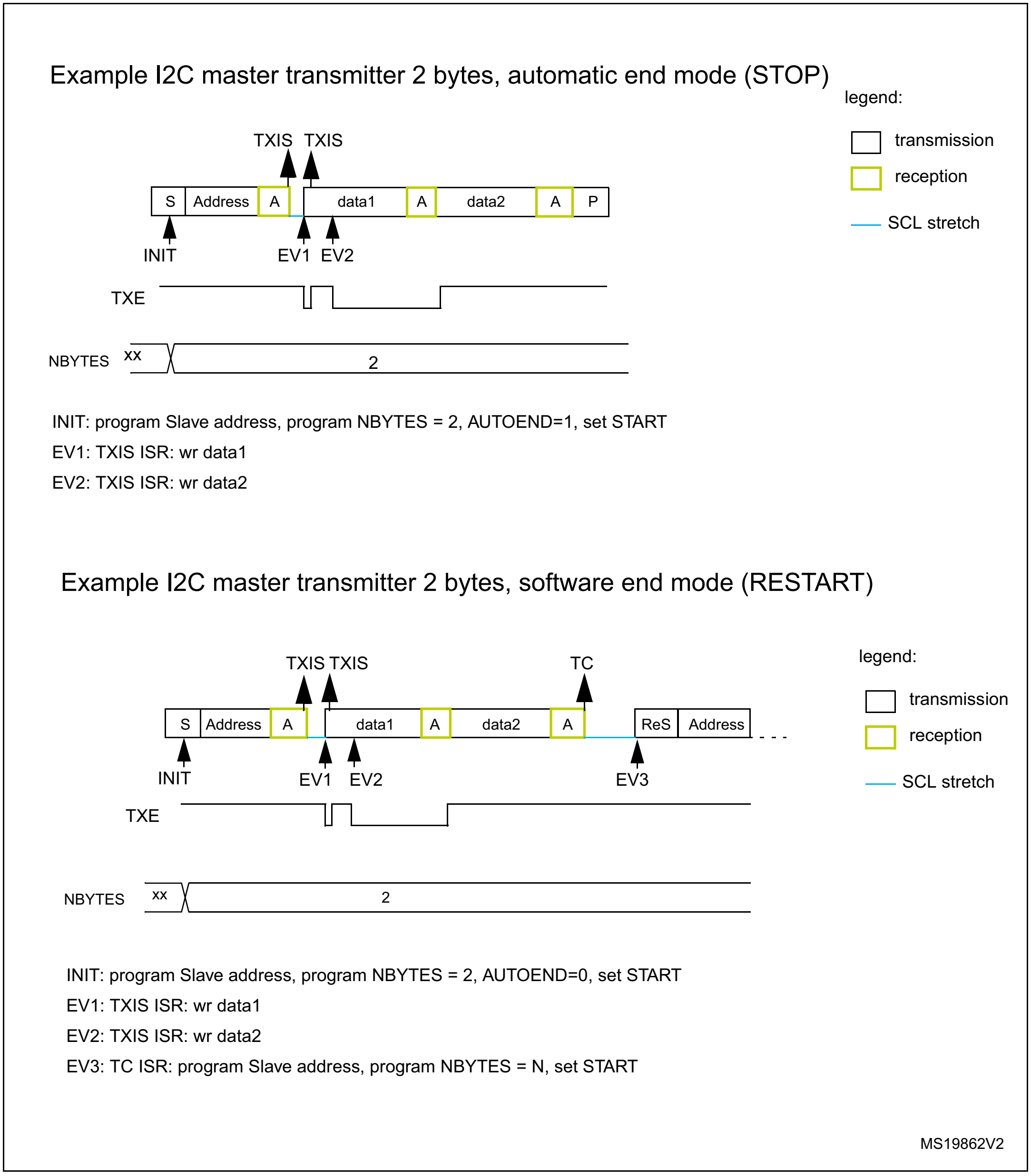


Figure 648. Transfer bus diagrams for I2C master transmitter



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

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

Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th 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 649. Transfer sequence flowchart for I2C master receiver for N≤255 bytes

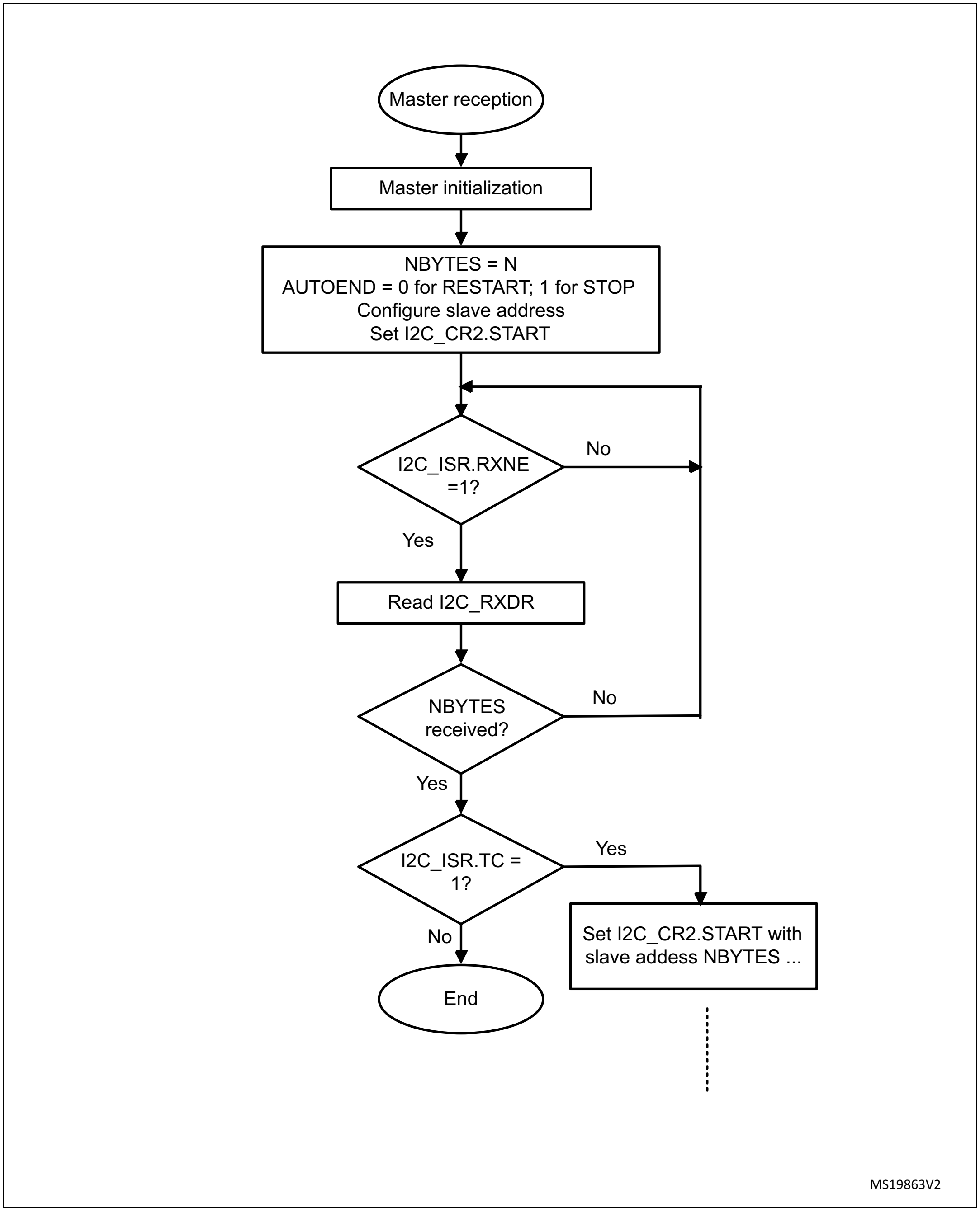


Figure 650. Transfer sequence flowchart for I2C master receiver for N >255 bytes

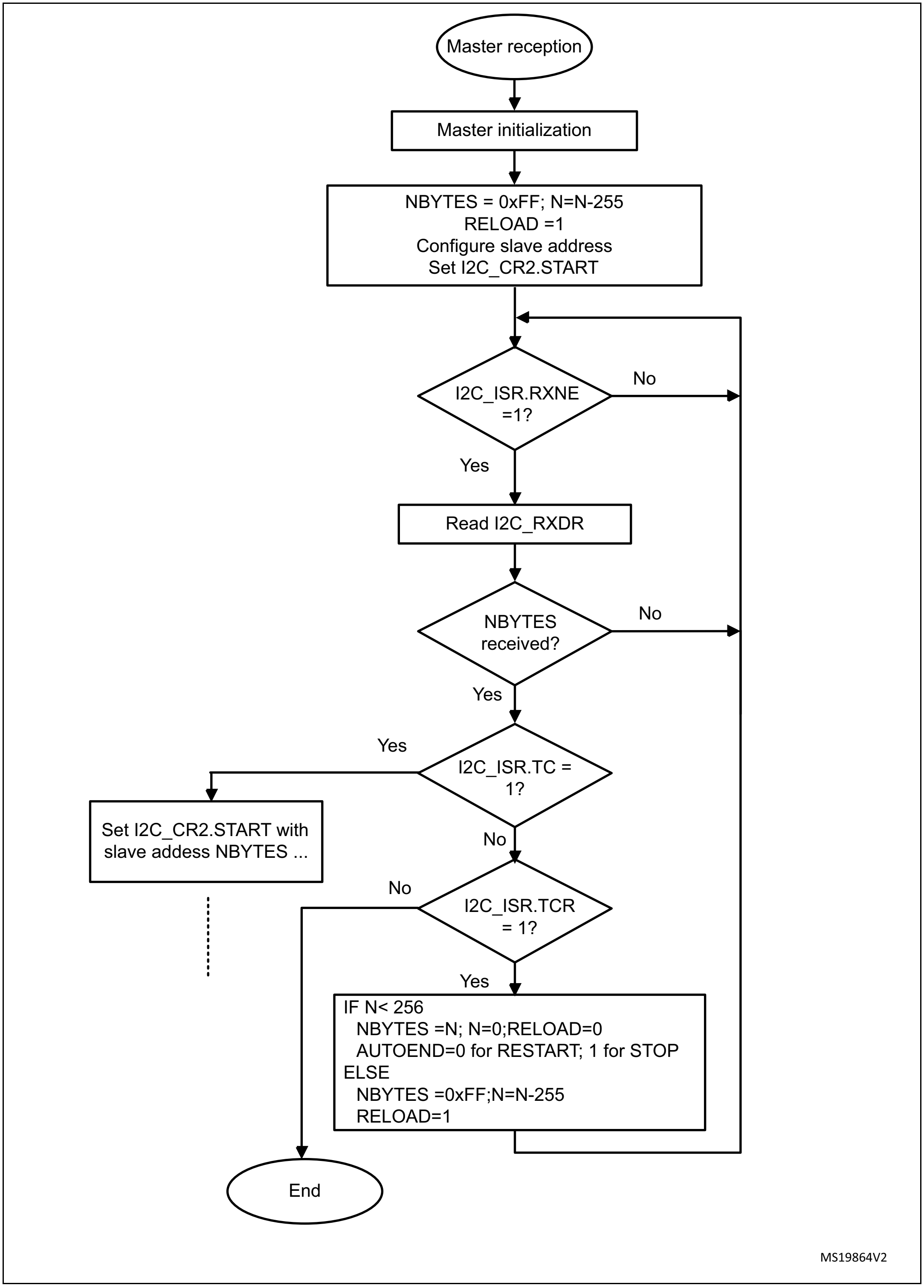
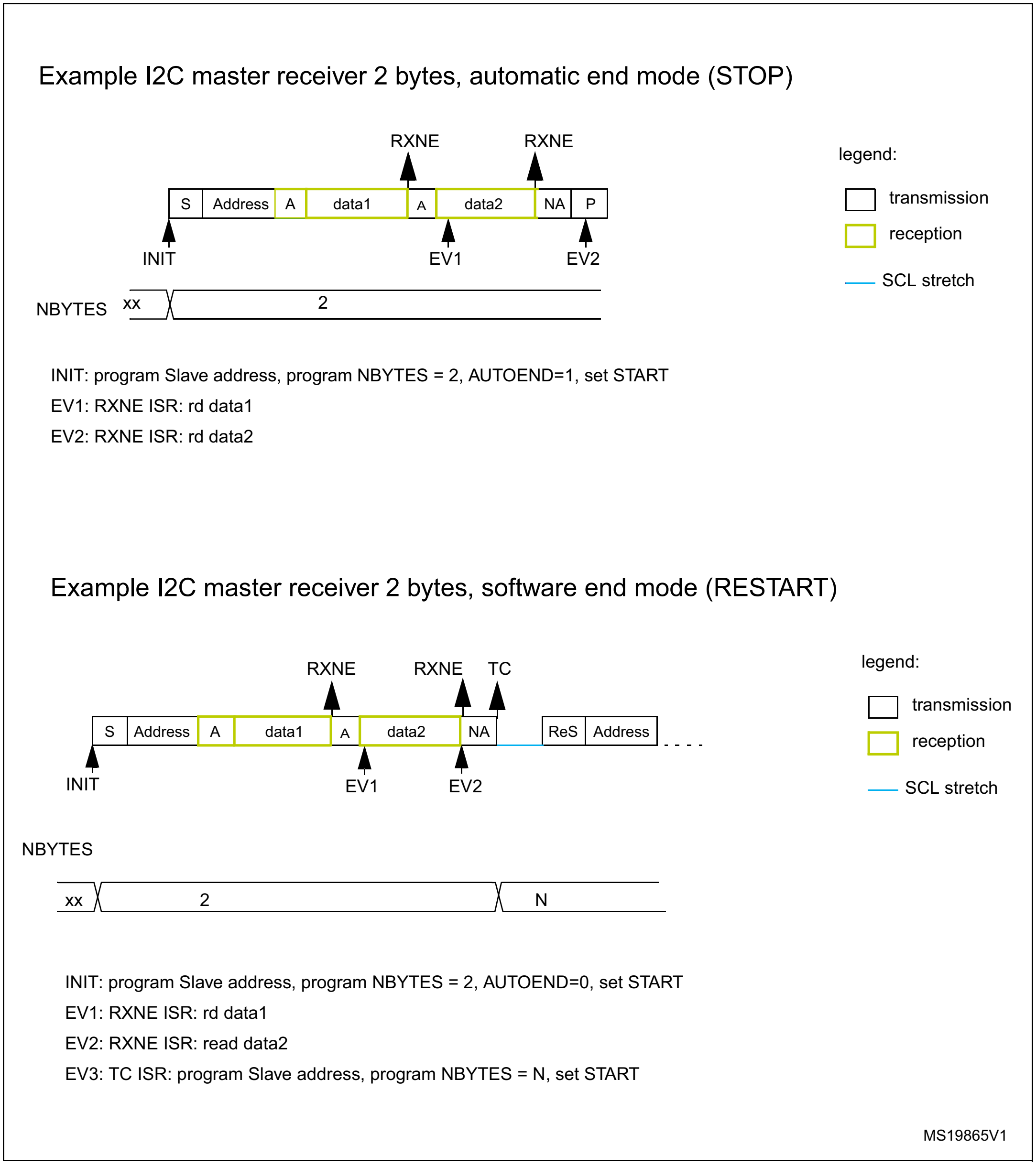


Figure 651. Transfer bus diagrams for I2C master receiver



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START

EV1: RXNE ISR: rd data1

EV2: RXNE ISR: rd data2

INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START

EV1: RXNE ISR: rd data1

EV2: RXNE ISR: read data2

EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19865V1

41.4.10 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I^{2}C specification. In order to get more accurate configuration values, the STM32CubeMX tool (I2C Configuration window) must be used.

Table 383. 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 384. 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.

Table 385. Examples of timings settings for f_{I2CCLK} = 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLH	0xC3	0xF	0x3	0x1
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~875 ns(4)
SDADEL	0x2	0x2	0x3	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 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. The SCL period t_{SCL} is greater than $t_{SCLL} + t_{SCLH}$ due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 1000 ns
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 750 ns
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 250 ns

41.4.11 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to *Section 41.3: 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 on page 1874* 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 3 timeouts defined in SMBus specification.

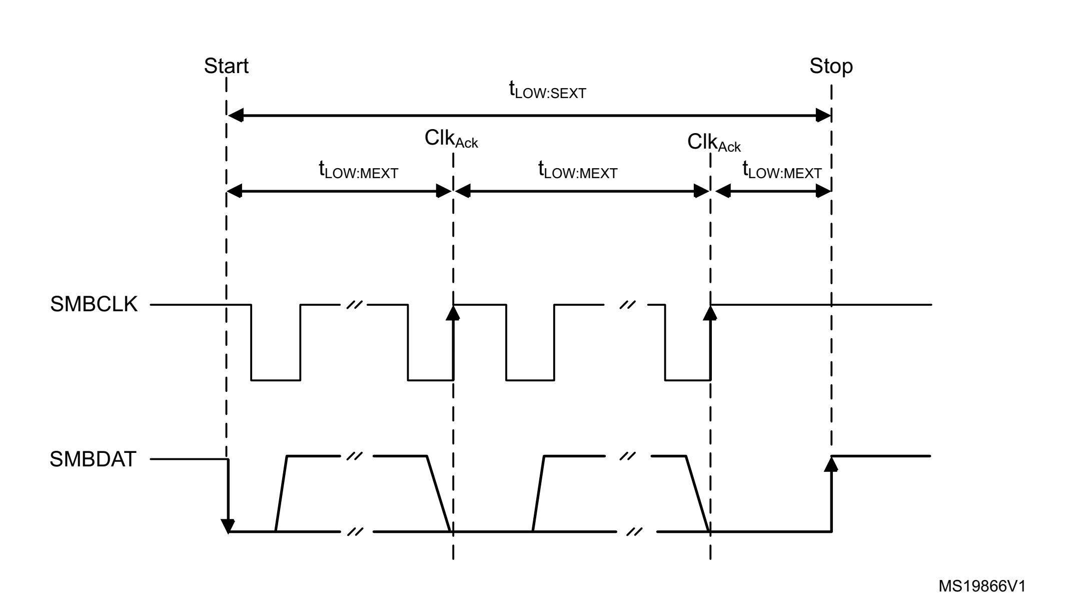
Table 386. SMBus timeout specifications

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	ms
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (master device)	-	10	ms

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 652. 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 380: I2C-SMBus specification data setup and hold times*)

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.

41.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to *Section 41.3: 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 1874* for more details.

Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to *Bus idle detection on page 1897* 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 387. 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 388: Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{TIMEOUT}$ = 25 ms)*.

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 on page 1897* section, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 389: Examples of TIMEOUTB settings for various I2CCLK frequencies*

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 390: Examples of TIMEOUTA settings for various I2CCLK frequencies (max t_{IDLE} = 50 µs)*

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

41.4.13 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 41.3: I2C implementation*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 388. 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
32 MHz	0x186	0	1	391 x 2048 x 31.25 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

- Configuring the maximum duration of $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ to 8 ms:

Table 389. 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
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 µs

Table 390. 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
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 µs

41.4.14 SMBus slave mode

This section is relevant only when the SMBus feature is supported. Refer to *Section 41.3: I2C implementation*.

In addition to I2C slave transfer management (refer to *Section 41.4.8: I2C slave mode*) some additional software flowcharts 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 653. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC

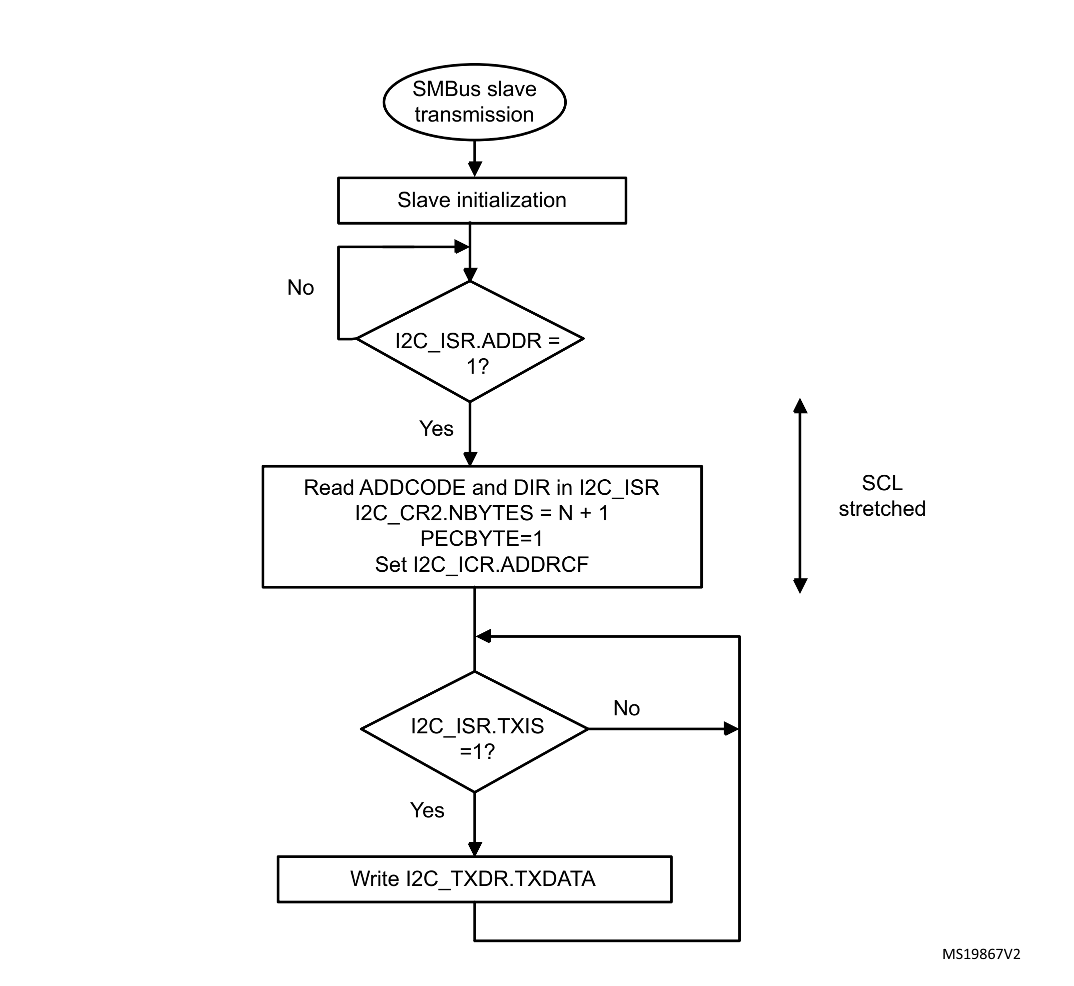
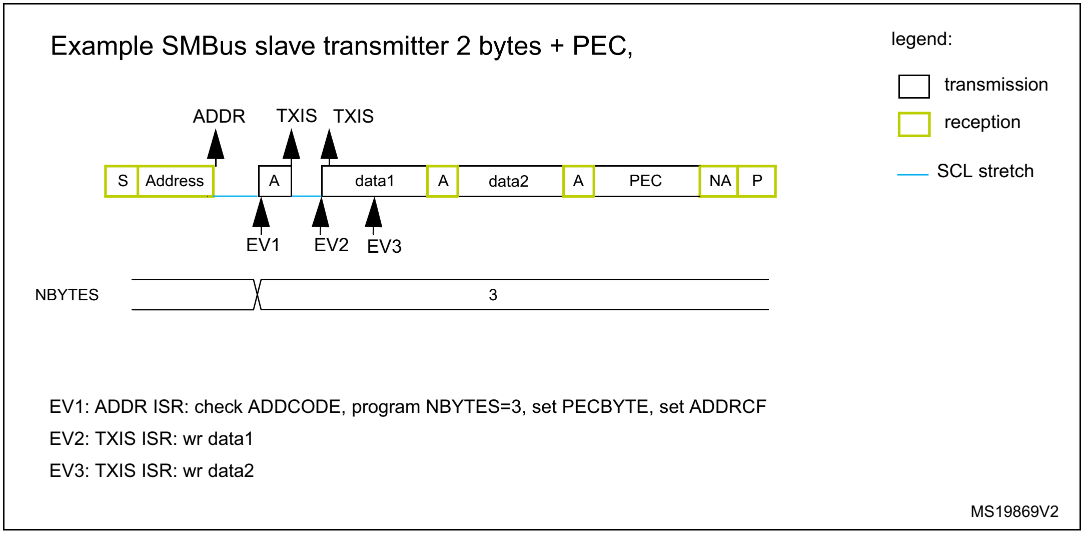


Figure 654. Transfer bus diagrams for SMBus slave transmitter (SBC=1)



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 on page 1874* 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 655. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

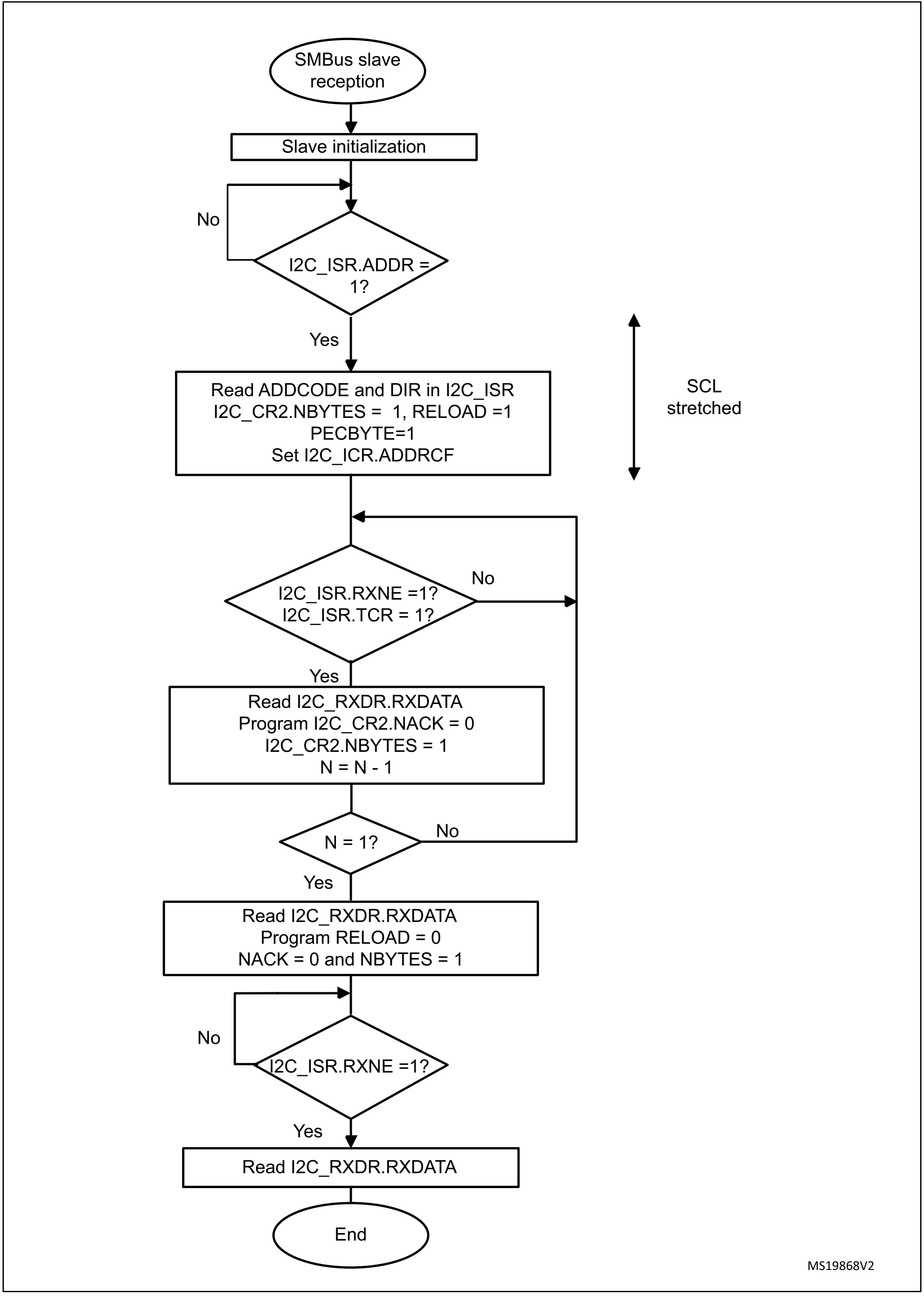
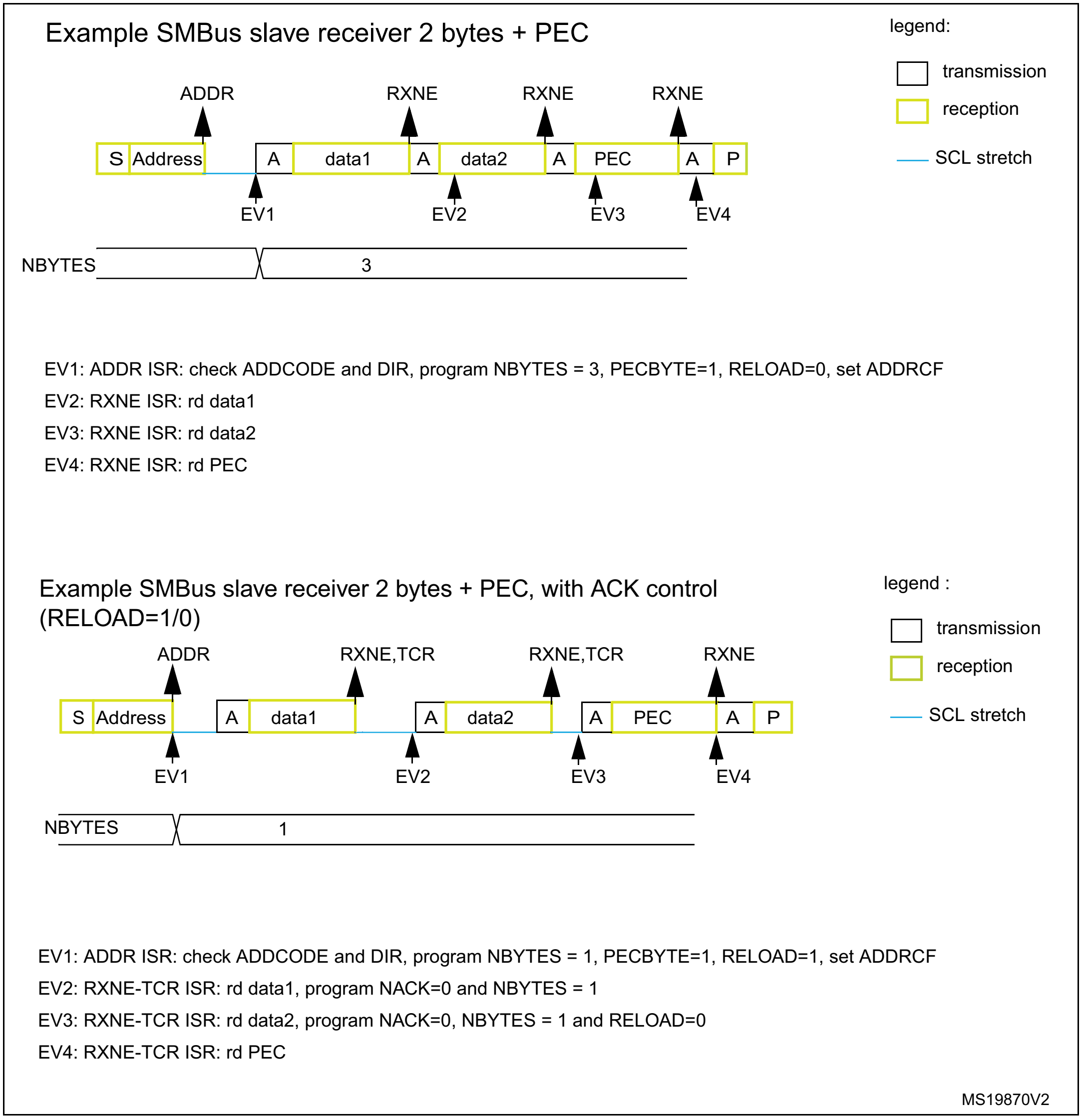


Figure 656. Bus transfer diagrams for SMBus slave receiver (SBC=1)



EV1: ADDR ISR: check ADDCODE and DIR, program NBYTES = 3, PECBYTE=1, RELOAD=0, set ADDRCF

EV2: RXNE ISR: rd data1

EV3: RXNE ISR: rd data2

EV4: RXNE ISR: rd PEC

EV1: ADDR ISR: check ADDCODE and DIR, program NBYTES = 1, PECBYTE=1, RELOAD=1, set ADDRCF

EV2: RXNE-TCR ISR: rd data1, program NACK=0 and NBYTES = 1

EV3: RXNE-TCR ISR: rd data2, program NACK=0, NBYTES = 1 and RELOAD=0

EV4: RXNE-TCR ISR: rd PEC

This section is relevant only when the SMBus feature is supported. Refer to *Section 41.3: I2C implementation*.

In addition to I2C master transfer management (refer to *Section 41.4.9: I2C master mode*) some additional software flowcharts are provided to support the SMBus.

SMBus master transmitter

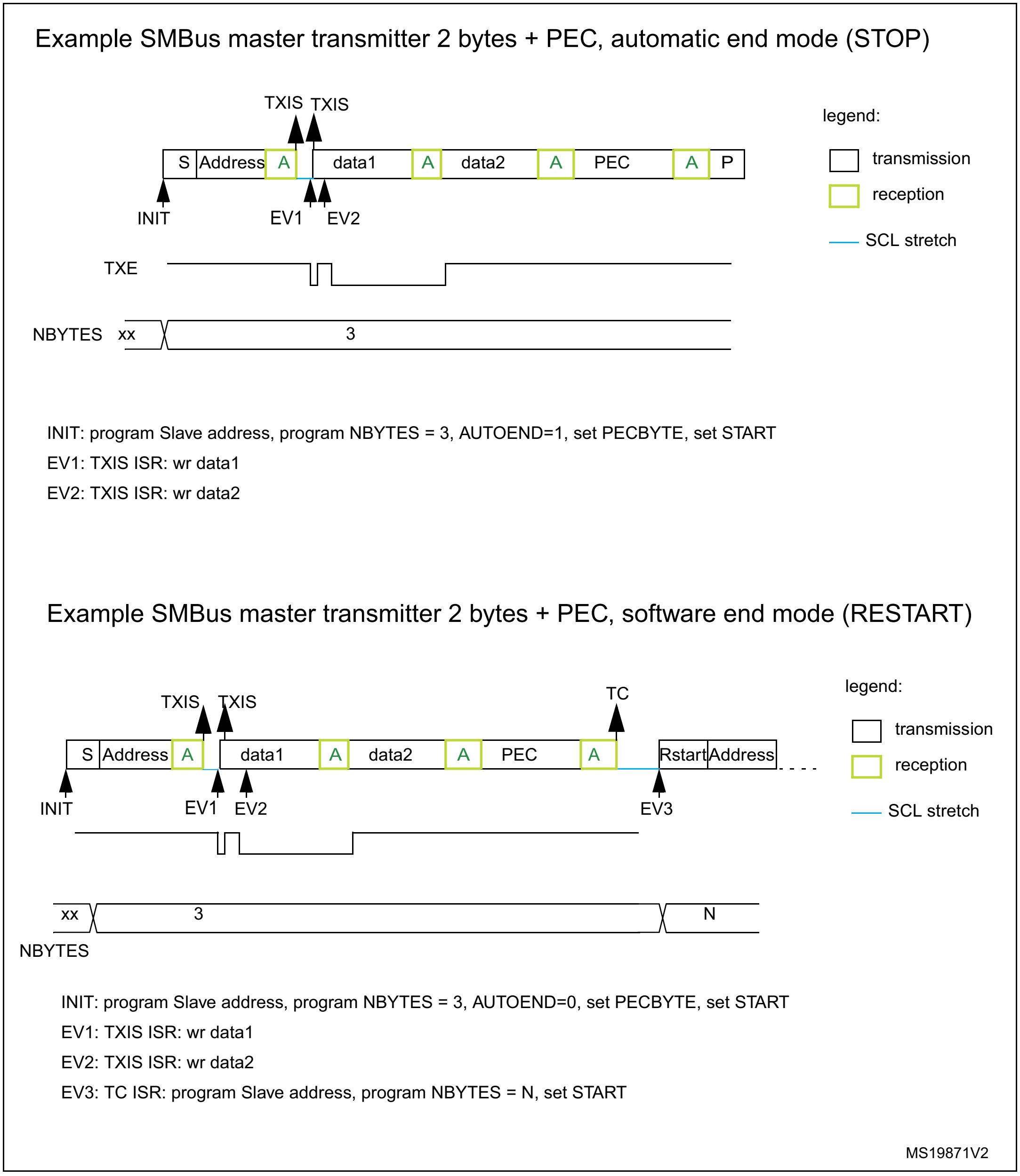
When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the 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 657. Bus transfer diagrams for SMBus master transmitter



INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19871V2

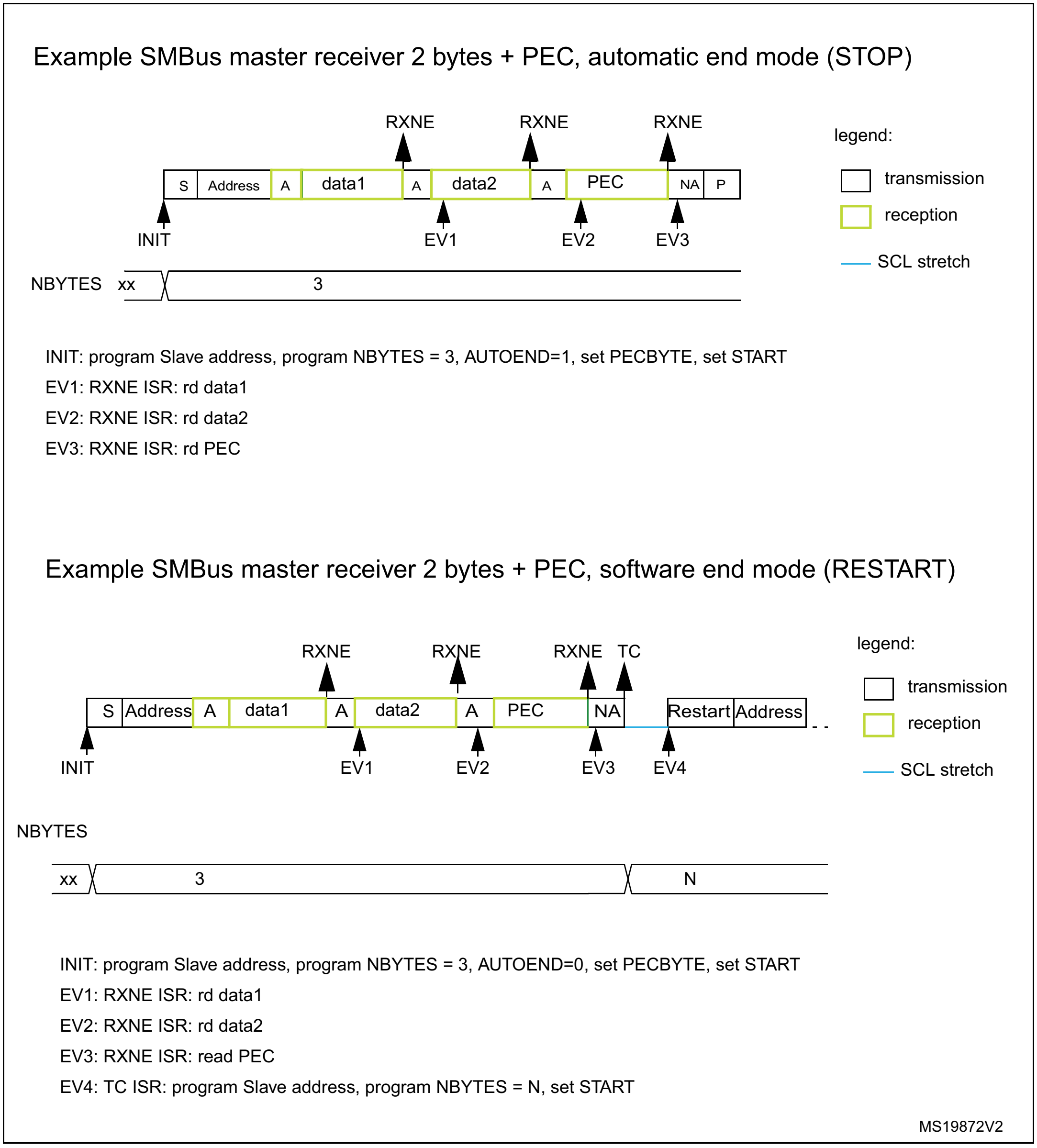
SMBus master receiver

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 658. 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

41.4.15 Wakeup from Stop mode on address match

This section is relevant only when wakeup from Stop mode feature is supported. Refer to *Section 41.3: 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).

41.4.16 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 41.3: 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 41.3: 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 41.3: 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.

41.4.17 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 14: Direct memory access controller (DMA) on page 481*) 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 *on page 1885*.
- 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 *on page 1900* and SMBus master transmitter *on page 1903*.

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 *Section 12: Direct memory access controller (DMA) on page 404*) 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 41.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to SMBus Slave receiver *on page 1901* and SMBus master receiver *on page 1905*.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

41.4.18 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.

41.5 I2C low-power modes

Table 391. 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. 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 41.3: I2C implementation* 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.

41.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 392. I2C Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the Standby modes
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 41.3: I2C implementation*.

41.7 I2C registers

Refer to Section 1.2 on page 73 for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

41.7.1 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 41.3: 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 41.3: 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 41.3: 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 41.3: 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 41.3: 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.

41.7.2 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.	PECBYTE	AUTOEND	RELOAD	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 41.3: 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 a timeout error detection, or when PE = 0. It can also be cleared by software by writing ‘1’ to the ADDRCF bit in the I2C_ICR register.

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.

41.7.3 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.

41.7.4 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.

41.7.5 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 1866*) and for SCL high and low level counters (refer to *I2C master initialization on page 1881*).

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.

41.7.6 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 41.3: I2C implementation.

41.7.7 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 41.3: 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 41.3: 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 41.3: 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.

41.7.8 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 41.3: 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 41.3: 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 41.3: 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.

41.7.9 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 41.3: I2C implementation.

41.7.10 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^{2}C bus

41.7.11 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^{2}C bus

Note: These bits can be written only when TXE=1.

41.7.12 I2C register map

The table below provides the I2C register map and reset values.

Table 393. 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.	ANOFF	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 393. 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 2.2 on page 81* for the register boundary addresses.

42 Independent watchdog (IWDG)

42.1 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 43 on page 1937*.

42.2 IWDG main features

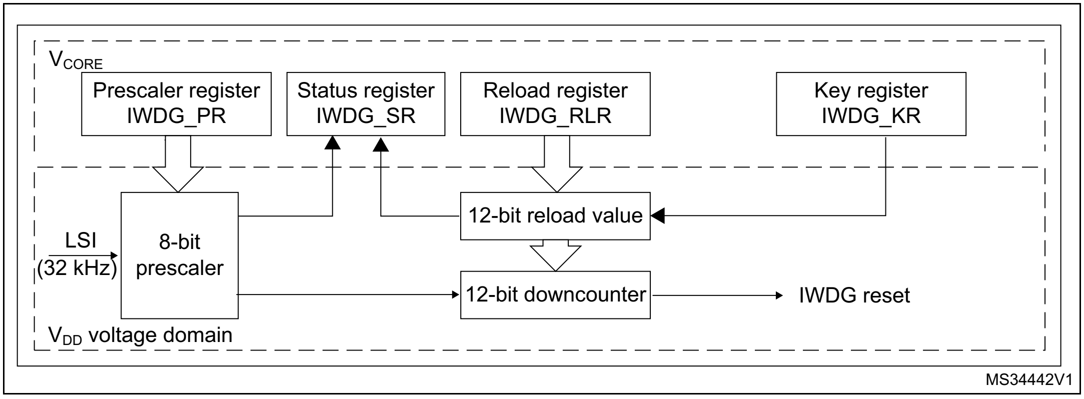
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

42.3 IWDG functional description

42.3.1 IWDG block diagram

Figure 659 shows the functional blocks of the independent watchdog module.

Figure 659. Independent watchdog block diagram



1. The register interface is located in the voltage domain. The watchdog function is located in the V_{DD} voltage domain, still functional in 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.

42.3.2 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).

42.3.3 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.

42.3.4 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.

42.3.5 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.

42.3.6 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.

42.4 IWDG registers

Refer to *Section 1.2 on page 73* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

42.4.1 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 42.3.5: Register access protection*)

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

42.4.2 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 42.3.5: 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.

42.4.3 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.

42.4.4 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.

42.4.5 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 42.3.5, 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.

42.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

Table 394. 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 2.2 on page 81* for the register boundary addresses.

43 System window watchdog (WWDG)

43.1 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.

43.2 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 661*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40.

43.3 WWDG functional description

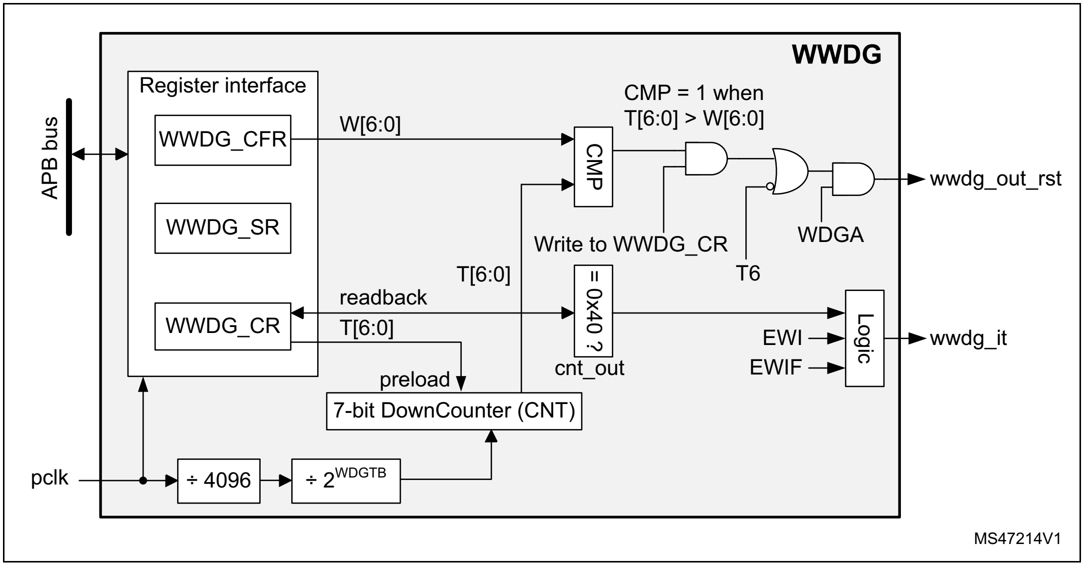
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, 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 660* for the WWDG block diagram.

43.3.1 WWDG block diagram

Figure 660. Watchdog block diagram



43.3.2 Enabling the watchdog

When the user option WWDG_SW selects “Software window 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.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

43.3.3 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 661*). 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 661* 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).

43.3.4 How to program the watchdog timeout

Use the formula in *Figure 661* 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.

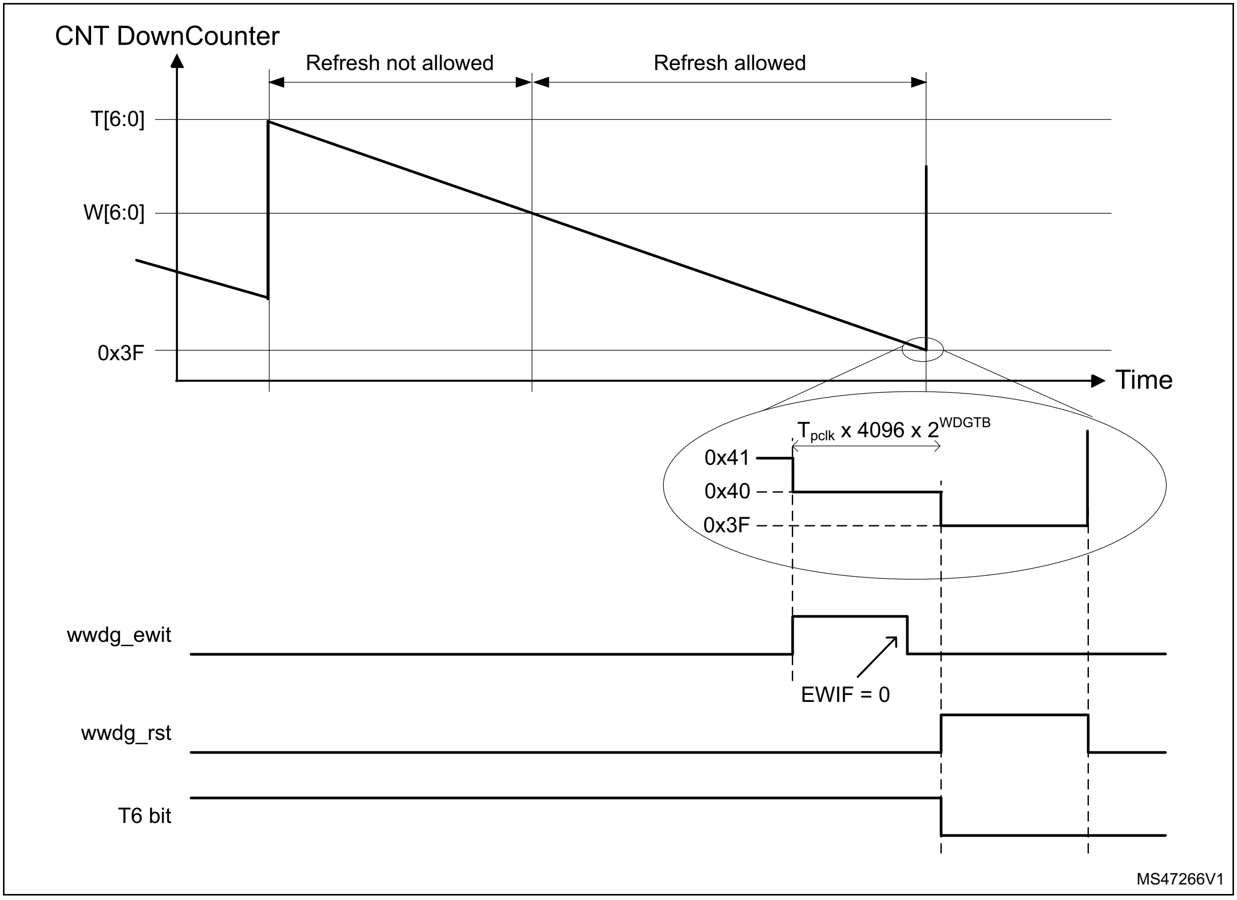


Figure 661. Window watchdog timing diagram

The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \qquad (ms)$$

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:

$$t_{WWDG} = (1/48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

43.3.5 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 48.16.2: Debug support for timers, RTC, watchdog, bxCAN and I²C*.

43.4 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 (e.g. due to a system lock in a higher priority task) the WWDG reset is eventually generated.

43.5 WWDG registers

Refer to *Section 1.2 on page 73* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

43.5.1 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 $2^{\text{WDGTB[2:0]}}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

43.5.2 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.

43.5.3 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.

43.5.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 395. 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 2.2 on page 81* for the register boundary addresses.

44 FD controller area network (FDCAN)

44.1 Introduction

The controller area network (CAN) subsystem (see *Figure 662*) consists of one CAN module, a shared message RAM and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

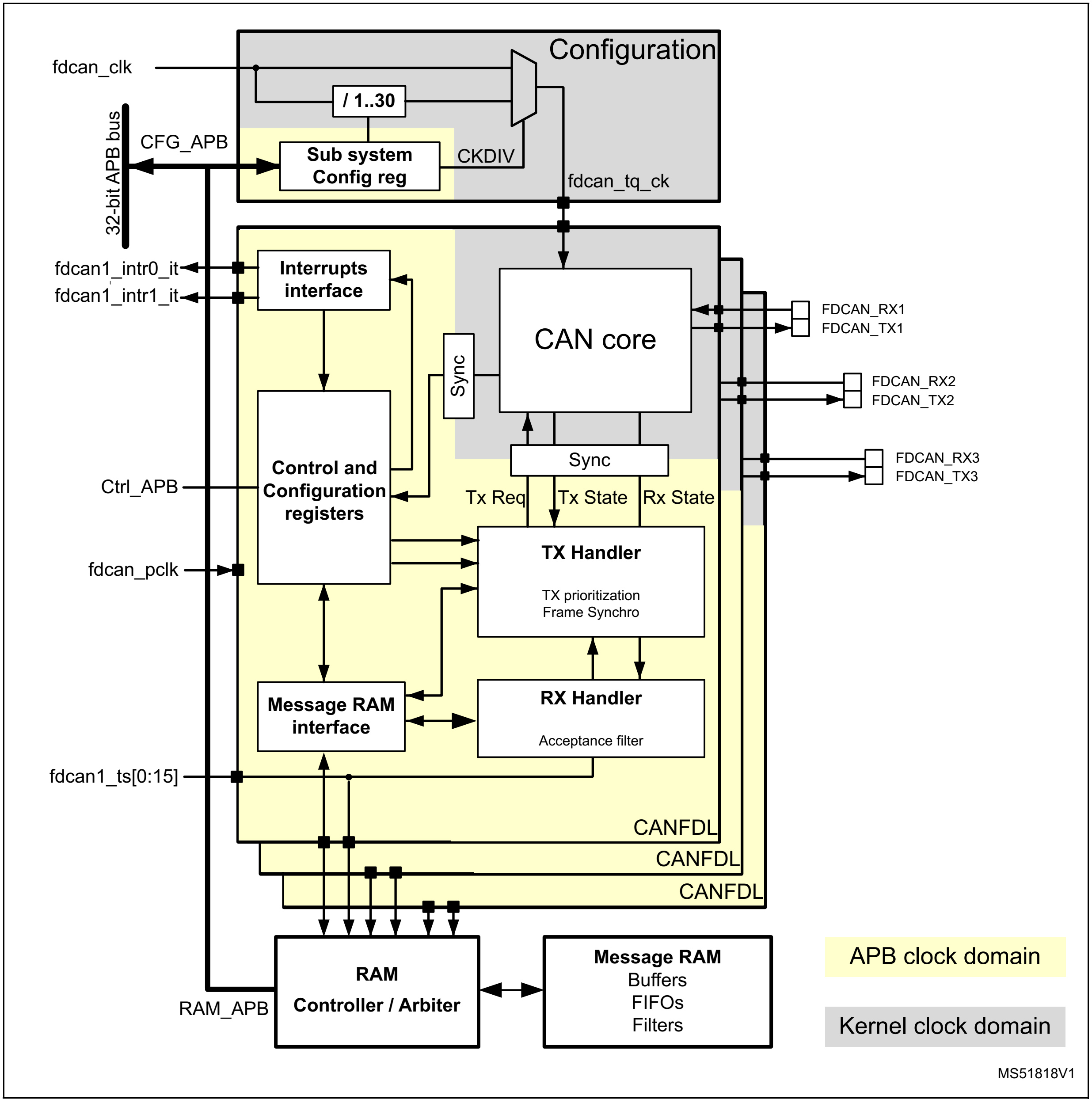
A 0.8-Kbyte message RAM per FDCAN instance implements filters, receive FIFOs, transmit event FIFOs and transmit FIFOs.

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 396* and *Figure 662*.

Table 396. CAN subsystem I/O signals

Name	Type	Description
fdcan_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdan_intr0_it	Digital output	FDCAN interrupt0
fdan_intr1_it		FDCAN interrupt1
fdcan_ts[0:15]	-	External timestamp vector
FDCAN_RX	Digital input	FDCAN receive pin
FDCAN_TX	Digital output	FDCAN transmit pin
APB interface	Digital input/output	Single APP with multiple psel for configuration, control and RAM access

Figure 662. CAN subsystem

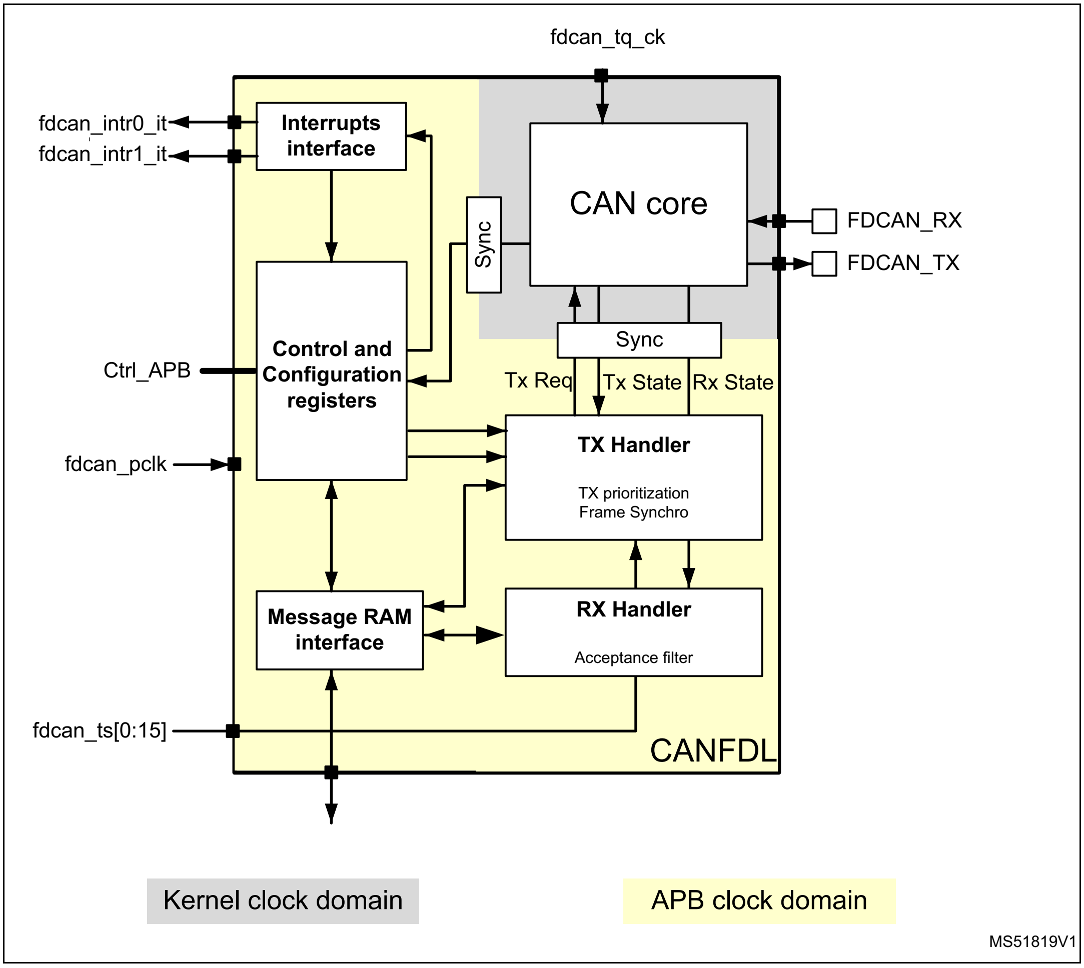


44.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015, -4
- CAN FD with maximum 64 data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable Transmit FIFO / queue of three payload (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power down support

44.3 FDCAN functional description

Figure 663. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it.

By programming EINT0 and EINT1 bits in FDCAN_ILE register, the interrupt lines can be separately enabled or disabled.

CAN core

The CAN core contains the protocol controller and receive / transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync

The Sync block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.

Tx handler

Controls the message transfer from the Message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. Tx buffer can be used as Tx FIFO or a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.

Rx handler

Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs, up to 8 filters for 29-bit IDs.

APB interface

Connects the FDCAN to the APB bus for configuration registers, controller configuration and RAM access.

Message RAM interface

Connects the FDCAN access to an external 1 Kbyte Message RAM through a RAM controller / arbiter.

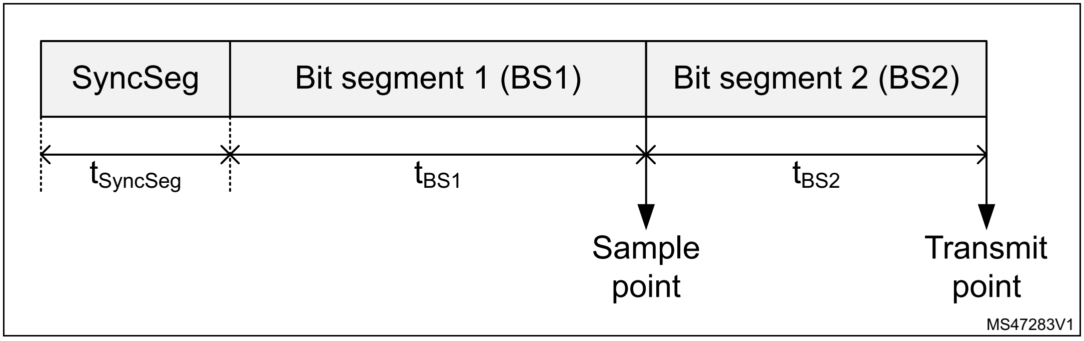
44.3.1 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in *Figure 664*, its operation may be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum (1 x tq).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta, but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but may also be automatically shortened to compensate for negative phase drifts.

Figure 664. Bit timing



The baud rate is the inverse of bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components. *Figure 664* indicates that bit time = $t_{SyncSeg}$ + t_{BS1} + t_{BS2}, where:

- for the nominal bit time
  - $tq = (FDCAN\_NBTP.NBRP[8:0] + 1) * t_{fdcan\_tq\_clk}$
  - $t_{SyncSeg} = 1\ tq$
  - $t_{BS1} = tq * (FDCAN\_NBTP.NTSEG1[7:0] + 1)$
  - $t_{BS2} = tq * (FDCAN\_NBTP.NTSEG2[6:0] + 1)$
- for the data bit time
  - $tq = (FDCAN\_DBTP.DBRP[4:0] + 1) * t_{fdcan\_tq\_clk}$
  - $t_{SyncSeg} = 1\ tq$
  - $t_{BS1} = tq * (FDCAN\_DBTP.DTSEG1[4:0] + 1)$
  - $t_{BS2} = tq * (FDCAN\_DBTP.DTSEG2[3:0] + 1)$

The (Re)Synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the Bit timing register is only possible while the device is in Standby mode. Registers FDCAN_DBTP and FDCAN_NBTP (dedicated, respectively, to data and nominal bit timing) are only accessible when CCCR.CCE and CCCR.INIT are set.

Note: For a detailed description of the CAN bit timing and resynchronization mechanism refer to the ISO 11898-1 standard.

44.3.2 Operating modes

Configuration

Access to IP version, hardware and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.

Software initialization

Software initialization is started by setting INIT bit in FDCAN_CCCR register, either by software or by a hardware reset, or by going Bus_Off. While INIT bit in FDCAN_CCCR register is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output FDCAN_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting INIT bit in FDCAN_CCCR does not change any configuration register. Clearing INIT bit in FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when both INIT bit in FDCAN_CCCR register and CCE bit in FDCAN_CCCR register are set.

CCE bit in FDCAN_CCCR register can only be set/cleared while INIT bit in FDCAN_CCCR is set. CCE bit in FDCAN_CCCR register is automatically cleared when INIT bit in FDCAN_CCCR is cleared.

The following registers are reset when CCE bit in FDCAN_CCCR register is set:

- FDCAN_HPMS - High priority message status
- FDCAN_RXF0S - Rx FIFO 0 status
- FDCAN_RXF1S - Rx FIFO 1 status
- FDCAN_TXFQS - Tx FIFO/Queue status
- FDCAN_TXBRP - Tx buffer request pending
- FDCAN_TXBTO - Tx buffer transmission occurred
- FDCAN_TXBCF - Tx buffer cancellation finished
- FDCAN_TXEFS - Tx event FIFO status

The timeout counter value TOC bit in FDCAN_TOCV register is preset to the value configured by TOP bit in FDCAN_TOCC register when CCE bit in FDCAN_CCCR is set.

In addition the state machines of the Tx Handler and Rx handler are held in idle state while CCE bit in FDCAN_CCCR is set.

The following registers can be written only when CCE bit in FDCAN_CCCR register is cleared:

- TXBAR - Tx buffer add request
- TXBCR - Tx buffer cancellation request

TEST bit in FDCAN_CCCR and MON bit in FDCAN_CCCR can only be set by software while both INIT bit in CCCR and CCE bit in CCCR register are set. Both bits may be reset at any time. DAR bit in FDCAN_CCCR can only be set/cleared while both INIT bit in FDCAN_CCCR and CCE bit in FDCAN_CCCR are set.

Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT Operation Mode is not supported.

Once the FDCAN is initialized and INIT bit in FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, Tx FIFO or Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the FDCAN protocol:

1. Long frame mode (LFM), where the data field of a CAN frame may be longer that eight bytes
2. Fast frame mode (FFM), where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame

Fast Frame Mode can be used in combination with Long Frame Mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a Protocol exception event by setting bit PSR.PXE. When Protocol exception handling is enabled (CCCR.PXHD = 0), this causes the operation state to change from Receiver (PSR.ACT = 10) to Integrating (PSR.ACT = 00) at the next sample point. In case Protocol exception Handling is disabled (CCCR.PXHD = 1), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming CCCR.FDOE. In case CCCR.FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via bit FDF in the respective Tx buffer element. With CCCR.FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx buffer element is set. CCCR.FDOE and CCCR.BRSE can only be changed while CCCR.INIT and CCCR.CCE are both set.

With CCCR.FDOE = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With CCCR.FDOE = 1 and CCCR.BRSE = 0, only bit FDF of a Tx buffer element is evaluated. With CCCR.FDOE = 1 and CCCR.BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming is completed. Then all nodes switch back to Classic CAN communication.

In the FDCAN format, the coding of the DLC differs from the one of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 397*.

Table 397. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD Fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the Bit timing and prescaler register BTP. In the following FDCAN data phase, the fast CAN bit timing is used as defined by the Fast bit timing and prescaler register FBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with a FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (tq), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case a FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of a FDCAN transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of a FDCAN frame is limited by the transceivers loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit DBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq (minimum time quantum, that is one period of fdcan_tq_ck clock).

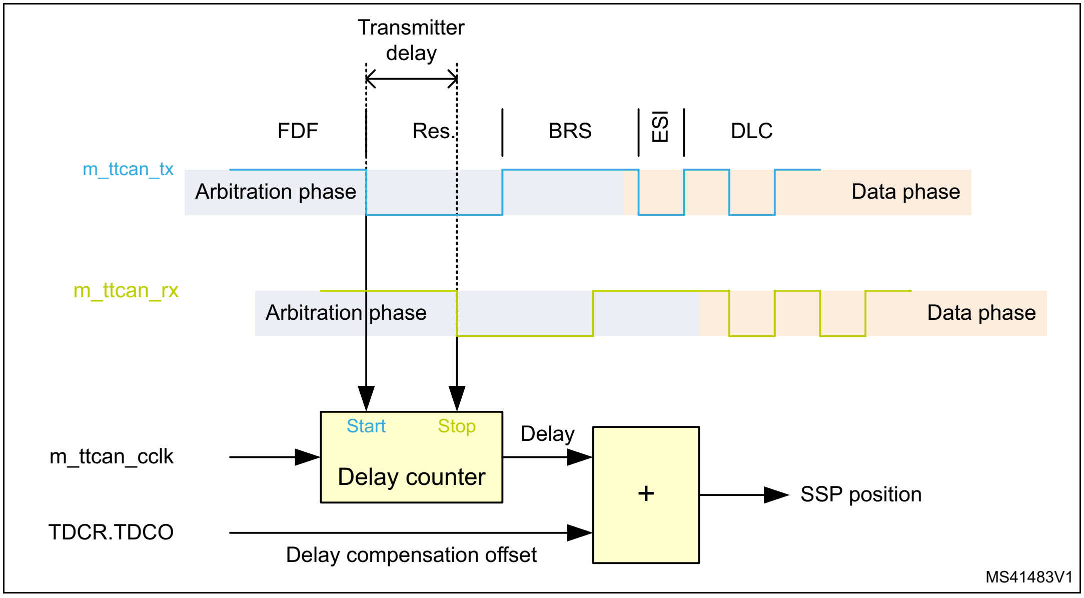
PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_Tx to FDCAN_Rx and the configured transmitter delay compensation offset TDCR.TDCO has to be lower than 6 bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset TDCR.TDCO has to be lower than or equal to 127 mtq. If the sum exceeds this value, the maximum value (127 mtq) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by programming DBTP.TDC = 1, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mtq.

Figure 665. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a to early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on FDCAN_RX, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCR.TDCF and FDCAN_RX is low.

Restricted operation mode

In Restricted operation mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (ECR.REC, ECR.TEC) are frozen while error logging (ECR.CEL) is active. The software can set the FDCAN into Restricted operation mode by setting bit CCCR.ASM. The bit can only be set by software when both CCCR.CCE and CCCR.INIT are set to 1. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted operation mode the software has to reset CCCR.ASM.

The Restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted operation mode after it has received a valid frame.

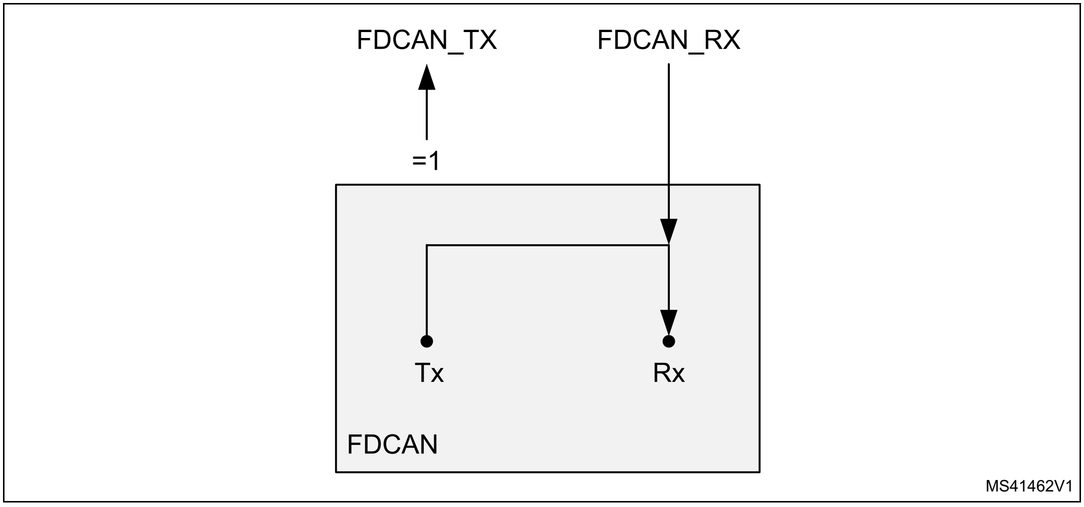
Note: The Restricted operation mode must not be combined with the Loop Back mode (internal or external).

Bus monitoring mode

The FDCAN is set in Bus monitoring mode by setting CCCR.MON bit. In Bus monitoring mode (for more details refer to ISO11898-1, 10.12 Bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In Bus monitoring mode the TXBRP register is held in reset state.

The Bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 666* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in Bus monitoring mode.

Figure 666. Pin control in Bus monitoring mode



Disabled automatic retransmission (DAR) mode

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled.

Frame transmission in DAR mode

In DAR mode all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx Request Pending bit TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, or has been aborted due to lost arbitration, or when an error has occurred during frame transmission.

- Successful transmission
  - Corresponding Tx buffer transmission occurred bit TXBTO[TOx] set
  - Corresponding Tx buffer cancellation finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation
  - Corresponding Tx buffer transmission occurred bit TXBTO[TOx] set
  - Corresponding Tx buffer cancellation finished bit TXBCF[CFx] set
- Arbitration loss or frame transmission disturbed
  - Corresponding Tx buffer transmission occurred bit TXBTO[TOx] not set
  - Corresponding Tx buffer cancellation finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with Event Type ET = 10 (transmission in spite of cancellation).

Power down (Sleep mode)

The FDCAN can be set into power down mode controlled by clock stop request input via CC control register CCCR[CSR]. As long as the clock stop request is active, bit CCCR[CSR] is read as 1.

When all pending transmission requests have completed, the FDCAN waits until bus idle state is detected. Then the FDCAN sets then CCCR[INIT] to 1 to prevent any further CAN transfers. Now the FDCAN acknowledges that it is ready for power down by setting CCCR[CSA] to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] has no effect. Now the module clock inputs may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC control register flag CCCR.CSR. The FDCAN acknowledges this by resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

Test modes

To enable write access to *FDCAN test register (FDCAN_TEST)*, bit CCCR.TEST must be set to 1, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming TEST.TX. In addition to its default function (the serial data output) it can drive the CAN Sample Point signal to monitor the FDCAN bit timing and it can drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from TEST.RX. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there may be a delay of several APB clock periods between writing to TEST.TX until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via TEST.RX.

Note: Test modes must be used for production tests or self test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External Loop Back mode

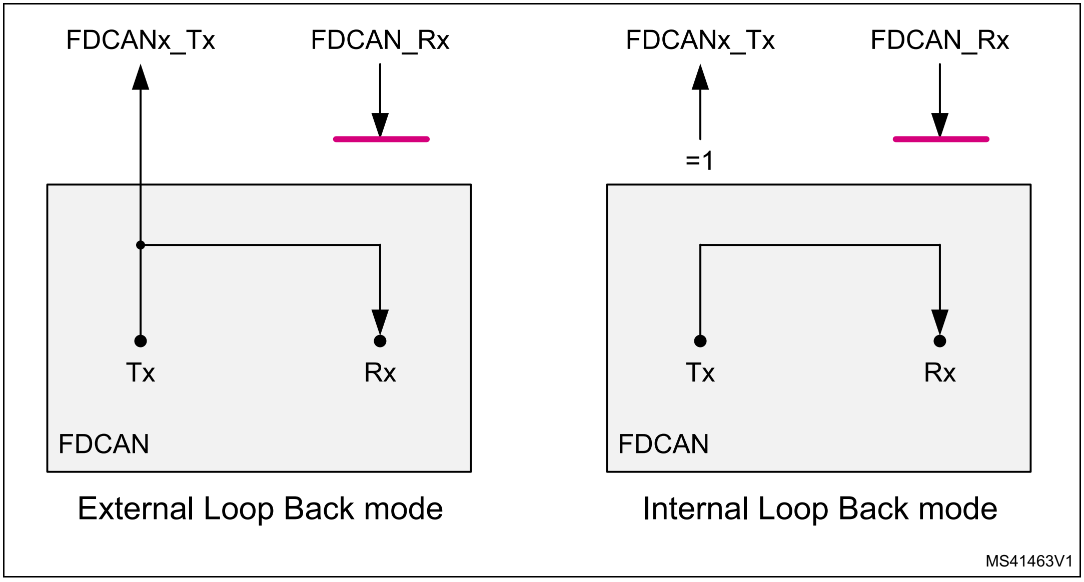
The FDCAN can be set in External Loop Back mode by programming TEST.LBCK to 1. In Loop Back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 667* shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data / remote frame) in Loop Back mode. In this mode the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST.LBCK and CCCR.MON to 1. This mode can be used for a “Hot Selftest”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. *Figure 667* shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of Internal Loop Back mode.

Figure 667. Pin control in Loop Back mode



Timestamp generation

For timestamp generation the FDCAN supplies a 16-bit wrap-around counter. A prescaler TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 ... 16). The counter is readable via TSCV[TCV]. A write access to register TSCV resets the counter to 0. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception/transmission the counter value is captured and stored into the timestamp section of a Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS, a 16-bit timestamp can be used.

Debug mode behavior

in debug mode the set / reset on read feature is automatically disabled during the debugger register access and enabled during normal MCU operation

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as downcounter and uses the same prescaler controlled by TSCC[TCP] as the Timestamp Counter. The timeout counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The timeout counter can only be started while CCCR[INIT] = 0. It is stopped when CCCR[INIT] = 1, e.g. when the FDCAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC[TOP] and continues downcounting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Downcounting is started when the first FIFO element is stored. Writing to TOCV has no effect.

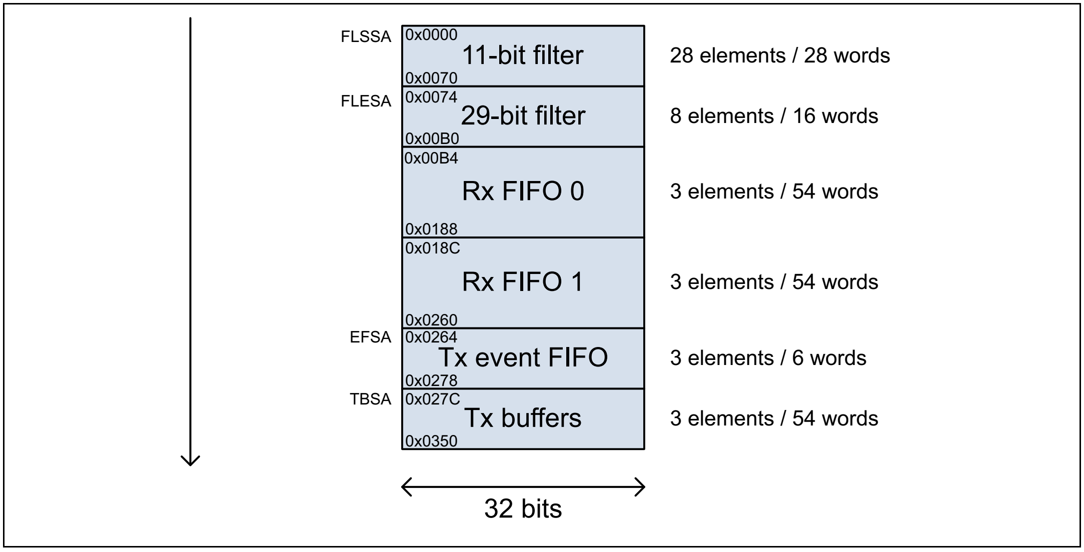
When the counter reaches 0, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore the point in time where the timeout counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in arbitration and data fields.

44.3.3 Message RAM

The Message RAM has a width of 32 bits, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in *Figure 668*.

Figure 668. Message RAM configuration



When the FDCAN addresses the Message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances the RAM start address for the FDCANn is computed by end address + 4 of FDCANn-1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:

- FDCAN1:
  - start address 0x0000
  - end address 0x0350 (as in *Figure 668*)
- FDCAN2:
  - start address = 0x0350 (FDCAN1 end address) + 4 = 0x0354
  - end address = 0x0354 (FDCAN2 start address) + 0x0350 - 4 = 0x06A0.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to Rx to one of the two Rx FIFOs, as well as the Rx FIFO Put and Get Indexes.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. Following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled/disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (RXGFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set High priority message interrupt flag IR[HPM]
- Set High priority message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1.

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the Message Handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact:

- **Rx FIFO**
  Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR.LEC and PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO Overwrite Mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filter(s). The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

Range filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT = 00: The Message ID of received frames is AND-ed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = 11: The Extended ID AND Mask (XIDAM) is not used for range filtering

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A 0 bit at the filter mask masks out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

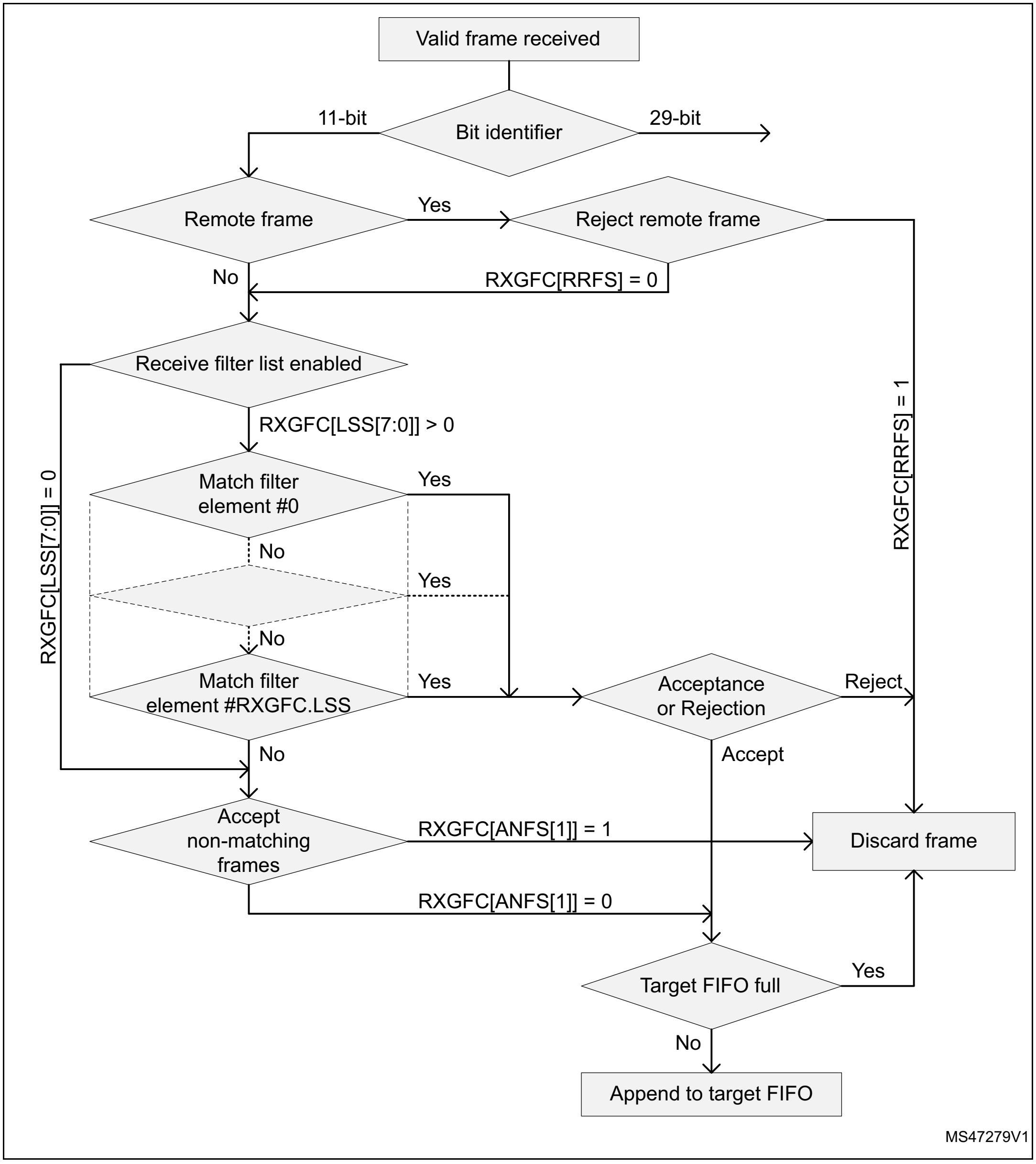
In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are 0, all Message IDs match.

Standard message ID filtering

Figure 669 shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID filter element is described in *Section 44.3.8*.

Controlled by the Global Filter Configuration (RXGFC) Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 669. Standard Message ID filter path

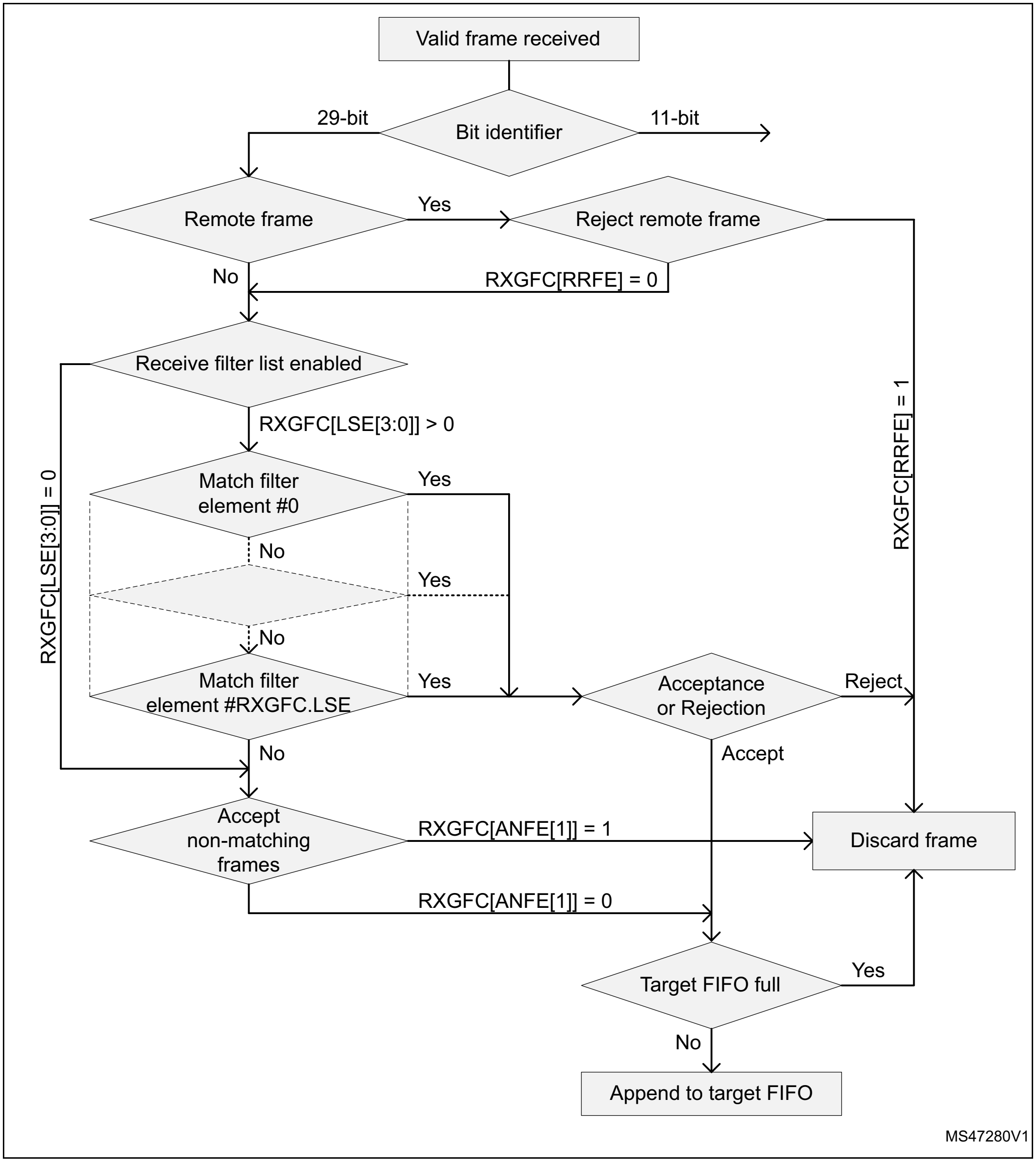


Extended message ID filtering

Figure 670 shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID filter element is described in *Section 44.3.9*.

Controlled by the Global Filter Configuration RXGFC and the Extended ID Filter Configuration RXGFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 670. Extended Message ID filter path



The Extended ID AND Mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx FIFO element is described in *Section 44.3.5*.

When an Rx FIFO full condition is signaled by IR[RFnF], no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and interrupt flag IR[RFnL] is set.

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS[FnGI] + FIFO Element Size has to be added to the corresponding Rx FIFO start address [FnSA].

Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXGFC.FnOM = 0. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signaled by RXFnS.FnF = 1. In addition interrupt flag IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled by RXFnS.RFnL = 1. In addition interrupt flag IR.RFnL is set.

Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXGFC.FnOM = 1.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signaled by RXFnS.FnF = 1, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements must start at least at get index + 1. This is because it can happen that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = 0).

Tx handling

The Tx Handler handles transmission requests for the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the Put and Get Indices, and the Tx event FIFO.Up to three Tx buffers can be set up for message transmission. The CAN message data field is configured to 64 bytes, Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

Table 398. Possible configurations for Frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bit rate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest Message ID) when the Tx buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If, as an example, CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by TXP bit in CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is disabled (CCCR.TXP = 0).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx

FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx buffers in the order these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF]= 1) is signaled. In this case no further messages must be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing 1 to the TXBAR bit related to the Tx buffer referenced by the Tx FIFO Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx buffer referenced by the Get Index is canceled, the Get Index is incremented to the next Tx buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx FIFO buffer is calculated by adding four times the Put Index TXFQS[TFQPI] (0 … 2) to the Tx buffer Start Address TBSA.

Tx queue

Tx queue operation is configured by programming TXBC[TFQM] to 1. Messages stored in the Tx queue are transmitted starting with the message with the lowest Message ID (highest priority).

In case of mixing of standard and extended Message IDs, the standard Message IDs are compared to bits [28:18] of extended Message IDs.

In case that multiple queue buffers are configured with the same Message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx buffer. In case that the Tx queue is full (TXFQS[TFQF] = 1), the Put Index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application may use register TXBRP instead of the Put Index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx queue buffer is calculated by adding four times the Tx queue Put Index TXFQS[TFQPI] (0 ... 2) to the Tx buffer Start Address TBSA.

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer the Host has to write a 1 to the corresponding bit position (= number of Tx buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to 1.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before it could have been started, there is a short time window where no transmission is started even if another message is pending in the node. This may enable another node to transmit a message that may have a priority lower than that of the second message in the node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the Message Marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in *Tx FIFO*.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by IR[TEFF], no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and interrupt flag IR[TEFL] is set.

When reading from the Tx event FIFO, two times the Tx event FIFO Get Index TXEFS[EFGI] has to be added to the Tx event FIFO start address EFSA.

44.3.4 FIFO acknowledge handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index, see *Section 44.4.23* and *Section 44.4.25*. Writing to the FIFO acknowledge index sets the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

1. When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.
2. When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO Get Index.

Due to the fact that the CPU has free access to the FDCAN Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High priority message from one of the two Rx FIFOs. In this case the FIFO Acknowledge Index must not be written because this would set the Get Index to a wrong position and also alters the FIFO Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The FDCAN does not check for erroneous values.

44.3.5 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the Message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in *Table 399*, the description is provided in *Table 400*.

Table 399. Rx FIFO element

<table>
<tr><th>Bit</th><th colspan="4">31 24</th><th colspan="4">23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>R0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="7">ID[28:0]</td></tr>
<tr><td>R1</td><td>ANMF</td><td colspan="3">FIDX[6:0]</td><td>Res.</td><td>FDF</td><td>BRS</td><td>DLC[3:0]</td><td colspan="2">RXTS[15:0]</td></tr>
<tr><td>R2</td><td colspan="4">DB3[7:0]</td><td colspan="4">DB2[7:0]</td><td>DB1[7:0]</td><td>D[7:0]</td></tr>
<tr><td>R3</td><td colspan="4">DB7[7:0]</td><td colspan="4">DB6[7:0]</td><td>DB5[7:0]</td><td>DB4[7:0]</td></tr>
<tr><td>⋮</td><td colspan="4">⋮</td><td colspan="4">⋮</td><td colspan="2">⋮</td></tr>
<tr><td>Rn</td><td colspan="4">DBm[7:0]</td><td colspan="4">DBm-1[7:0]</td><td>DBm-2[7:0]</td><td>DBm-3[7:0]</td></tr>
</table>

The element size configured for storage of CAN FD messages is set to 64 bytes data field.

Table 400. Rx FIFO element description

Field	Description
R0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
R0 Bit 30 XTD	Extended identifier Signals to the Host whether the received frame has a standard or extended identifier. – 0: 11-bit standard identifier – 1: 29-bit extended identifier
R0 Bit 29 RTR	Remote transmission request Signals to the Host whether the received frame is a data frame or a remote frame. – 0: Received frame is a data frame – 1: Received frame is a remote frame
R0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

Table 400. Rx FIFO element description (continued)

Field	Description
R1 Bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames may be enabled via RXGFC[ANFS] and RXGFC[ANFE]. – 0: Received frame matching filter index FIDX – 1: Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]	Filter index 0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range is 0 to RXGFC[LSS] - 1 or RXGFC[LSE] - 1.
R1 Bit 21 FDF	FD format – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
R1 Bit 20 BRS	Bit rate switch – 0: Frame received without bit rate switching – 1: Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]	Data length code – 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9-15: Classic CAN: received frame has 8 data bytes – 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0 RXTS[15:0]	Rx timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 Bits 31:24 DB3[7:0]	Data Byte 3
R2 Bits 23:16 DB2[7:0]	Data Byte 2
R2 Bits 15:8 DB1[7:0]	Data Byte 1
R2 Bits 7:0 D[7:0]	Data Byte 0
R3 Bits 31:24 DB7[7:0]	Data Byte 7
R3 Bits 23:16 DB6[7:0]	Data Byte 6
R3 Bits 15:8 DB5[7:0]	Data Byte 5
R3 Bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮
Rn Bits 31:24 DBm[7:0]	Data Byte m

Table 400. Rx FIFO element description (continued)

Field	Description
Rn Bits 23:16 DBm-1[7:0]	Data Byte m-1
Rn Bits 15:8 DBm-2[7:0]	Data Byte m-2
Rn Bits 7:0 DBm-3[7:0]	Data Byte m-3

44.3.6 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The Tx Handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration FDCAN_TXBC.TFQM . The element size is configured for storage of CAN FD messages with up to 64 bytes data.

Table 401. Tx buffer and FIFO element

<table>
<tr><th>Bit</th><th colspan="4">31 24</th><th colspan="5">23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>T0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="8">ID[28:0]</td></tr>
<tr><td>T1</td><td colspan="4">MM[7:0]</td><td>EFC</td><td>Res.</td><td>FDF</td><td>BPS</td><td>DLC[3:0]</td><td colspan="2">Res.</td></tr>
<tr><td>T2</td><td colspan="4">DB3[7:0]</td><td colspan="5">DB2[7:0]</td><td>DB1[7:0]</td><td>D[7:0]</td></tr>
<tr><td>T3</td><td colspan="4">DB7[7:0]</td><td colspan="5">DB6[7:0]</td><td>DB5[7:0]</td><td>DB4[7:0]</td></tr>
<tr><td>⋮</td><td colspan="4">⋮</td><td colspan="5">⋮</td><td colspan="2">⋮</td></tr>
<tr><td>Tn</td><td colspan="4">DBm[7:0]</td><td colspan="5">DBm-1[7:0]</td><td>DBm-2[7:0]</td><td>DBm-3[7:0]</td></tr>
</table>

Table 402. Tx buffer element description

Field	Description
T0 Bit 31 ESI(1)	Error state indicator – 0: ESI bit in CAN FD format depends only on error passive flag – 1: ESI bit in CAN FD format transmitted recessive
T0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
T0 Bit 29 RTR(2)	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
T0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.

Table 402. Tx buffer element description (continued)

Field	Description
T1 Bit 23 EFC	Event FIFO control – 0: Do not store Tx events – 1: Store Tx events
T1 Bit 21 FDF	FD format – 0: Frame transmitted in Classic CAN format – 1: Frame transmitted in CAN FD format
T1 Bit 20 BRS(3)	Bit rate switching – 0: CAN FD frames transmitted without bit rate switching – 1: CAN FD frames transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code – 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9 - 15: Classic CAN: received frame has 8 data bytes – 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24 DB3[7:0]	Data Byte 3
T2 Bits 23:16 DB2[7:0]	Data Byte 2
T2 Bits 15:8 DB1[7:0]	Data Byte 1
T2 Bits 7:0 D[7:0]	Data Byte 0
T3 Bits 31:24 DB7[7:0]	Data Byte 7
T3 Bits 23:16 DB6[7:0]	Data Byte 6
T3 Bits 15:8 DB5[7:0]	Data Byte 5
T3 Bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮
Tn Bits 31:24 DBm[7:0]	Data Byte m
Tn Bits 23:16 DBm-1[7:0]	Data Byte m-1
Tn Bits 15:8 DBm-2[7:0]	Data Byte m-2
Tn Bits 7:0 DBm-3[7:0]	Data Byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node always transmits the ESI bit recessive.

2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if CCCR.FDOE enables the transmission in CAN FD format.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled CCCR.FDOE = 1. Bit BRS is only evaluated when in addition CCCR.BRSE = 1.

44.3.7 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx event FIFO can be obtained from register TXEFS.

Table 403. Tx event FIFO element

<table>
<tr><th>Bit</th><th colspan="4">31 24</th><th colspan="4">23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>E0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="7">ID[28:0]</td></tr>
<tr><td>E1</td><td colspan="4">MM[7:0]</td><td>ET[1:0]</td><td>EDL</td><td>BRS</td><td>DLC[3:0]</td><td colspan="2">TXTS[15:0]</td></tr>
</table>

Table 404. Tx event FIFO element description

Field	Description
E0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 Bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 Bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.
E1 Bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 Bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit rate switching – 0: Frame transmitted without bit rate switching – 1: Frame transmitted with bit rate switching

Table 404. Tx event FIFO element description (continued)

Field	Description
T1 Bits 19:16 DLC[3:0]	Data length code 0 - 8: Frame with 0-8 data bytes transmitted 9 - 15: Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

44.3.8 FDCAN Standard message ID Filter element

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address FLSSA plus the index of the filter element (0 … 27).

Table 405. Standard Message ID Filter element

<table>
<tr><th>Bit</th><th colspan="3">31 24</th><th>23 16</th><th colspan="2">15 8</th><th>7 0</th></tr>
<tr><td>S0</td><td>SFT[1:0]</td><td>SFEC[2:0]</td><td colspan="2">SFID1[10:0]</td><td>Res.</td><td colspan="2">SFID2[10:0]</td></tr>
</table>

Table 406. Standard Message ID Filter element Field description

Field	Description
Bit 31:30 SFT[1:0][1]	Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]	Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = 100, 101 or 110 a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
Bits 26:16 SFID1[10:0]	Standard filter ID 1 First ID of standard ID filter element.
Bits 10:0 SFID2[10:0]	Standard filter ID 2 Second ID of standard ID filter element.

1. With SFT = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC = 000).

Note: In case a reserved value is configured, the filter element is considered disabled.

44.3.9 FDCAN Extended message ID filter element

Up to 8 filters element can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address FLESA plus two times the index of the filter element (0 … 7).

Table 407. Extended Message ID Filter element

Bit	31	24	23 16	15 8	7 0
F0	EFEC[2:0]		EFID1[28:0]		
F1	EFTI[1:0]	Res.	EFID2[28:0]		

Table 408. Extended Message ID Filter element field description

Field	Description
F0 Bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
F0 Bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism.
F1 Bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID >= EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID >= EF1ID), XIDAM mask not applied
F1 Bit 29	Not used
F1 Bits 28:0 EFID2[28:0]	Extended filter ID 2 Second ID of extended ID filter element.

44.4 FDCAN registers

44.4.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
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
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: 3

Bits 27:24 **STEP[3:0]**: 2

Bits 23:20 **SUBSTEP[3:0]**: 1

Bits 19:16 **YEAR[3:0]**: 4

Bits 15:8 **MON[7:0]**: 12

Bits 7:0 **DAY[7:0]**: 18

44.4.2 FDCAN endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[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
ETV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ETV[31:0]**: Endianness test value

The endianness test value is 0x8765 4321.

Note: The register read must give the reset value to ensure no endiandess issue.

44.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is only writable if bits CCCR.CCE and CCCR.INIT are set. The CAN time quantum may be programmed in the range of 1 to 32 FDCAN clock periods. tq = (DBRP + 1) FDCAN clock period.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [DTSEG1 + DTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								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.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			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 **TDC**: Transceiver delay compensation

0: Transceiver delay compensation disabled

1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bit rate prescaler

The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The hardware interpreters this value as the value programmed plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point

Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS1} = (DTSEG1 + 1) x tq.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point

Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS2} = (DTSEG2 + 1) x tq.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width

Must always be smaller than DTSEG2, valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1: t_{SJW} = (DSJW + 1) x tq.

Note: With a FDCAN clock of 8 MHz, the value of 0x00300A33 configures the FDCAN for a fast bit rate of 500 kbit/s.

44.4.4 FDCAN test register (FDCAN_TEST)

Write access to the Test register has to be enabled by setting bit CCCR[TEST] to 1. All Test register functions are set to their reset values when bit CCCR[TEST] is reset.

Loop Back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX differently from 00 may disturb the message transfer on the CAN bus.

Address offset: 0x0010

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.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

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

Bit 7 **RX**: Receive pin

Monitors the actual value of pin FDCANx_RX
0: The CAN bus is dominant (FDCANx_RX = 0)
1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time
01: Sample point can be monitored at pin FDCANx_TX
10: Dominant (0) level at pin FDCANx_TX
11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop back mode

0: Reset value, Loop Back mode is disabled
1: Loop Back mode is enabled (see *Power down (Sleep mode)*)

Bits 3:0 Reserved, must be kept at reset value.

44.4.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access starts the Message RAM Watchdog Counter with the value configured by the RWD[WDC] bits.

The counter is reloaded with RWD[WDC] bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message

RAM until the counter has counted down to 0, the counter stops and interrupt flag IR[WDI] bit is set. The RAM Watchdog Counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

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**
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00, the counter is disabled.
These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of FDCAN_CCCR register are set to 1.

44.4.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and resetting of single bits, see *Software initialization*.

31	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**
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non ISO operation

If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.

0: CAN FD frame format according to ISO11898-1

1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP:**

If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.

0: disabled

1: enabled

Bit 13 **EFBI**: Edge filtering during bus integration

0: Edge filtering disabled

1: Two consecutive dominant tq required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable

0: Protocol exception handling enabled

1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bit rate switching

0: Bit rate switching for transmissions disabled

1: Bit rate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable

0: FD operation disabled

1: FD operation enabled

Bit 7 **TEST**: Test mode enable

0: Normal operation, register TEST holds reset values

1: Test Mode, write access to register TEST enabled

Bit 6 **DAR**: Disable automatic retransmission

0: Automatic retransmission of messages not transmitted successfully enabled

1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode

Bit MON can only be set by software when both CCE and INIT are set to 1. The bit can be reset by the Host at any time.

0: Bus monitoring mode disabled

1: Bus monitoring mode enabled

Bit 4 **CSR**: Clock stop request

0: No clock stop requested

1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus reached idle.

Bit 3 **CSA**: Clock stop acknowledge

0: No clock stop acknowledged

1: FDCAN may be set in power down by stopping APB clock and kernel clock.

Bit 2 **ASM**: ASM restricted operation mode

The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the Restricted operation Mode after it has received a valid frame. In the optional Restricted operation Mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set to 1. The bit can be reset by the software at any time.

0: Normal CAN operation

1: Restricted operation Mode active

Bit 1 **CCE**: Configuration change enable

0: The CPU has no write access to the protected configuration registers.

1: The CPU has write access to the protected configuration registers (while CCCR.INIT = 1).

Bit 0 **INIT**: Initialization

0: Normal operation

1: Initialization started

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

44.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programed in the range of 4 to 81 tq. The CAN time quantum may be programmed in the range of [1 … 1024] FDCAN kernel clock periods.

tq = (BRP + 1) FDCAN clock period fdcan_clk

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
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
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that the used value is the one programmed incremented by one.

These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 24:16 **NBRP[8:0]**: Bit rate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN kernel clock of 48 MHz, the reset value of 0x06000A03 configures the FDCAN for a bit rate of 3 MBit/s.

44.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

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.	TCP[3:0]			
												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.	TSS[1:0]	
														rw	rw

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

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 … 16].

The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS = 10).

These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM3 value (tim3_cnt[0:15])

11: Same as 00.

These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

44.4.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

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
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When TSCC[TSS] = 01, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of TSCC[TCP]. A wrap around sets interrupt flag IR[TSW]. Write access resets the counter to 0.

When TSCC.TSS = 10, TSC reflects the external timestamp counter value. A write access has no impact.

Note: A “wrap around” is a change of the Timestamp Counter value from non-0 to 0 that is not caused by write access to TSCV.

44.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[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.	TOS[1:0]		ETOC
													rw	rw	rw

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in Continuous mode, a write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored.

00: Continuous operation

01: Timeout controlled by Tx event FIFO

10: Timeout controlled by Rx FIFO 0

11: Timeout controlled by Rx FIFO 1

These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 0 **ETOC**: Timeout counter enable

0: Timeout counter disabled

1: Timeout counter enabled

This is a protected write (P) bit, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

For more details see *Timeout counter*.

44.4.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

31	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**
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of TSCC.TCP. When decremented to 0, interrupt flag IR.TOO is set and the timeout counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

44.4.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

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.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR[ELO].

Access type is RX: reset on read.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128.

1: The receive error counter has reached the error passive level of 128.

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

44.4.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs	rs	rs	r	r	r	r	r	rs	rs	rs

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

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and TDCR.TDCO. The SSP position is, in the data phase, the number of minimum time quanta (mtq) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access

1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.

0: Since this bit was reset by the CPU, no FDCAN message has been received.

1: Message in FDCAN format with EDL flag set has been received.

Access type is RX: reset on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its BRS flag set.

1: Last received FDCAN message had its BRS flag set.

Access type is RX: reset on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its ESI flag set.

1: Last received FDCAN message had its ESI flag set.

Access type is RX: reset on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC. This field is cleared to 0 when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.

Access type is RS: set on read.

Bit 7 **BO**: Bus_Off status

0: The FDCAN is not Bus_Off.

1: The FDCAN is in Bus_Off state.

Bit 6 **EW**: Warning Sstatus

0: Both error counters are below the Error_Warning limit of 96.

1: At least one of error counter has reached the Error_Warning limit of 96.

Bit 5 **EP**: Error passive

0: The FDCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.

1: The FDCAN is in the Error_Passive state.

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module's CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication.

01: Idle: node is neither receiver nor transmitter.

10: Receiver: node is operating as receiver.

11: Transmitter: node is operating as transmitter.

Bits 2:0 **LEC[2:0]:** Last error code

The LEC indicates the type of the last error to occur on the CAN bus. This field is cleared to 0 when a message has been transferred (reception or transmission) without error.

000: No Error: No error occurred since LEC has been reset by successful reception or transmission.

001: Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form Error: A fixed format part of a received frame has the wrong format.

011: AckError: The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: NoChange: Any read access to the Protocol status register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol status register.

Access type is RS: set on read.

Note: When a frame in FDCAN format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) is shown in FLEC instead of LEC. An error in a fixed stuff bit of a FDCAN CRC sequence is shown as a Form Error, not Stuff Error.

Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT]. If the device goes Bus_Off, it sets CCCR.INIT of its own, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters are reset. During the waiting time after the reset of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

44.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

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.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 mtq.

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

44.4.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position.

Writing a 0 has no effect. A hard reset clears the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

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.	ARA	PED	PEA	WDI	BO	EW	EP	ELO
								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**
TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
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 31:24 Reserved, must be kept at reset value.

Bit 23 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 22 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (PSR.DLEC different from 0,7)

Bit 21 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (PSR.LEC different from 0,7)

Bit 20 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 19 **BO**: Bus_Off status
0: Bus_Off status unchanged
1: Bus_Off status changed

Bit 18 **EW**: Warning status
0: Error_Warning status unchanged
1: Error_Warning status changed

Bit 17 **EP**: Error passive
0: Error_Passive status unchanged
1: Error_Passive status changed

Bit 16 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow.
1: Overflow of CAN error logging counter occurred.

Bit 15 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 14 **MRAF**: Message RAM access failure
The flag is set when the Rx handler:

- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the FDCAN is switched into Restricted operation Mode (see *Restricted operation mode*). To leave Restricted operation Mode, the Host CPU has to reset CCCR.ASM.
0: No Message RAM access failure occurred
1: Message RAM access failure occurred

Bit 13 **TSW**: Timestamp wraparound
0: No timestamp counter wrap-around
1: Timestamp counter wrapped around

Bit 12 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost

Bit 11 **TEFF**: Tx event FIFO full
0: Tx event FIFO Not full
1: Tx event FIFO full

Bit 10 **TEFN**: Tx event FIFO New Entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element.

Bit 9 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 8 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 7 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 6 **HPM**: High-priority message
0: No high-priority message received
1: High-priority message received

Bit 5 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost

Bit 4 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 3 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 2 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost

Bit 1 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 0 **RF0N**: Rx FIFO 0 new message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

44.4.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register are signaled on an interrupt line.

Address offset: 0x0054

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.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
rw	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 **ARAE**: Access to reserved address enable

Bit 22 **PEDE**: Protocol error in data phase enable

Bit 21 **PEAE**: Protocol error in arbitration phase enable

Bit 20 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **BOE**: Bus_Off status
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **HPME**: High-priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

44.4.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific group of interrupt flag from the Interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via ILE[EINT0] and ILE[EINT1].

Address offset: 0x0058

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.	PERR	BERR	MISC	TFERR	SMSG	RxFIFO1	RxFIFO0
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERR:** Protocol error grouping the following interruption
ARAL: Access to reserved address line
PEDL: Protocol error in data phase line
PEAL: Protocol error in arbitration phase line
WDIL: Watchdog interrupt line
BOL: Bus_Off status
EWL: Warning status interrupt line

Bit 5 :**BERR:** Bit and line error grouping the following interruption
EPL Error passive interrupt line
ELOL: Error logging overflow interrupt line

Bit 4 **MISC:** Interrupt regrouping the following interruption
TOOL: Timeout occurred interrupt line
MRAFL: Message RAM access failure interrupt line
TSWL: Timestamp wraparound interrupt line

Bit 3 **TFERR:** Tx FIFO ERROR grouping the following interruption
TEFLL: Tx event FIFO element lost interrupt line
TEFFL: Tx event FIFO full interrupt line
TEFNL: Tx event FIFO new entry interrupt line
TFEL: Tx FIFO empty interrupt line

Bit 2 **SMSG:** Status message bit grouping the following interruption
TCFL: Transmission cancellation finished interrupt line
TCL: Transmission completed interrupt line
HPML: High-priority message interrupt line

Bit 1 **RxFIFO1:** RX FIFO bit grouping the following interruption
RF1LL: Rx FIFO 1 message lost interrupt line
RF1FL: Rx FIFO 1 full Interrupt line
RF1NL: Rx FIFO 1 new message interrupt line

Bit 0 **RxFIFO0:** RX FIFO bit grouping the following interruption
RF0LL: Rx FIFO 0 message lost interrupt line
RF0FL: Rx FIFO 0 full interrupt line
RF0NL: Rx FIFO 0 new message interrupt line

44.4.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming bits EINT0 and EINT1.

Address offset: 0x005C

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.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr1_it disabled
1: Interrupt line fdcan_intr1_it enabled

44.4.19 FDCAN global filter configuration register (FDCAN_RXGFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in *Figure 669* and *Figure 670*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSE[3:0]				Res.	Res.	Res.	LSS[4:0]				
				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.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
						rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **LSE[3:0]**: List size extended

0: No extended message ID filter
1 to 8: Number of extended message ID filter elements
>8: Values greater than 8 are interpreted as 8.
These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

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

Bits 20:16 **LSS[4:0]**: List size standard

0: No standard message ID filter
1 to 28: Number of standard message ID filter elements
>28: Values greater than 28 are interpreted as 28.
These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

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

Bit 9 **F0OM**: FIFO 0 operation mode (overwrite or blocking)

This is protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 8 **F1OM**: FIFO 1 operation mode (overwrite or blocking)

This is a protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept Non-matching frames standard

Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended

Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 1 **RRFS**: Reject remote frames standard

0: Filter remote frames with 11-bit standard IDs

1: Reject all remote frames with 11-bit standard IDs

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 0 **RRFE**: Reject remote frames extended

0: Filter remote frames with 29-bit standard IDs

1: Reject all remote frames with 29-bit standard IDs

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

44.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0084

Reset value: 0x1FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EIDM[28:16]												
			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**
EIDM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID mask

For acceptance filtering of extended frames the Extended ID AND Mask is AND-ed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to 1 the mask is not active.

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

44.4.21 FDCAN high-priority message status register (FDCAN_HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088

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**
FLST	Res.	Res.	FIDX[4:0]					MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
r			r	r	r	r	r	r	r				r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list

Indicates the filter list of the matching filter element.
0: Standard filter list
1: Extended filter list

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

Bits 12:8 **FIDX[4:0]**: Filter index

Index of matching filter element. Range is 0 to RXGFC[LSS] - 1 or RXGFC[LSE] - 1.

Bits 7:6 **MSI[1:0]**: Message storage indicator

00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]**: Buffer index

Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

44.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x0090

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.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]	
						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.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF0L**: Rx FIFO 0 message lost

This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full

0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]**: Rx FIFO 0 put index

Rx FIFO 0 write index pointer, range 0 to 2.

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

Bits 9:8 **F0GI[1:0]**: Rx FIFO 0 get index

Rx FIFO 0 read index pointer, range 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F0FL[3:0]**: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0, range 0 to 3.

44.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x0094

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.	F0AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index
After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This sets the Rx FIFO 0 get index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 fill level RXF0S[F0FL].

44.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x0098

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.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]	
						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.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message lost
This bit is a copy of interrupt flag IR[RF1L]. When IR[RF1L] is reset, this bit is also reset.
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 24 **F1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F1PI[1:0]**: Rx FIFO 1 put index
Rx FIFO 1 write index pointer, range 0 to 2.

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

Bits 9:8 **F1GI[1:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer, range 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F1FL[3:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1, range 0 to 3.

44.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x009C

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.	F1AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F1AI[2:0]**: Rx FIFO 1 acknowledge index
After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This sets the Rx FIFO 1 get index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].

44.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

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.	TFQM	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TFQM**: Tx FIFO/queue mode

0: Tx FIFO operation

1: Tx queue operation.

This is a protected write (P) bit, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 23:0 Reserved, must be kept at reset value.

44.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/Queue status is related to the pending Tx requests listed in register TXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (TXBRP not yet updated).

Address offset: 0x00C4

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.	TFQF	Res.	Res.	Res.	TFQPI[1:0]	
										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.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
						r	r						r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TFQF**: Tx FIFO/queue full

0: Tx FIFO/queue not full

1: Tx FIFO/queue full

Bits 20:18 Reserved, must be kept at reset value.

Bits 17:16 **TFQPI[1:0]**: Tx FIFO/queue put index

Tx FIFO/queue write index pointer, range 0 to 3

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

Bits 9:8 **TFGI[1:0]**: Tx FIFO get index

Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TXBC.TFQM = 1)

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

Bits 2:0 **TFFL[2:0]**: Tx FIFO free level

Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 3. Read as 0 when Tx queue operation is configured (TXBC[TFQM] = 1).

44.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00C8

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.	TRP[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TRP[2:0]**: Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been canceled via register TXBCR.

After a TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest Message ID).

A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled via TXBCF

after successful transmission together with the corresponding TXBTO bit

when the transmission has not yet been started at the point of cancellation

when the transmission has been aborted due to lost arbitration

when an error occurred during frame transmission

In DAR mode all transmissions are automatically canceled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this Add Request is canceled immediately, the corresponding TXBRP bit is reset.

44.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00CC

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.	AR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **AR[2:0]**: Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the Host to set transmission requests for multiple Tx buffers with one write to TXBAR. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding TXBRP bit already set), the request is ignored.

44.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D0

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.	CR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CR[2:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact.

This enables the Host to set cancellation requests for multiple Tx buffers with one write to TXBCR. The bits remain set until the corresponding TXBRP bit is reset.

0: No cancellation pending

1: Cancellation pending

44.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D4

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.	TO[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TO[2:0]**: Transmission occurred.

Each Tx buffer has its own TO bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register TXBAR.

0: No transmission occurred

1: Transmission occurred

44.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00D8

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.	CF[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CF[2:0]**: Cancellation finished

Each Tx buffer has its own CF bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register TXBAR.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

44.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00DC

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.	TIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TIE[2:0]**: Transmission interrupt enable
Each Tx buffer has its own TIE bit.
0: Transmission interrupt disabled
1: Transmission interrupt enable

44.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_ TXBCIE)

Address offset: 0x00E0

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.	CFIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CFIE[2:0]**: Cancellation finished interrupt enable.
Each Tx buffer has its own CFIE bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

44.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00E4

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.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]	
						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.	EFGI[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
						r	r						r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element lost

This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset.

0 No Tx event FIFO element lost

1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full

0: Tx event FIFO not full

1: Tx event FIFO full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **EFPI[1:0]**: Event FIFO put index

Tx event FIFO write index pointer, range 0 to 3.

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

Bits 9:8 **EFGI[1:0]**: Event FIFO get index

Tx event FIFO read index pointer, range 0 to 3.

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

Bits 2:0 **EFFL[2:0]**: Event FIFO fill level

Number of elements stored in Tx event FIFO, range 0 to 3.

44.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00E8

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.	EFAI[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **EFAI[1:0]**: Event FIFO acknowledge index

After the Host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI. This sets the Tx event FIFO get index TXEFS[EFGI] to EFAI + 1 and updates the FIFO 0 fill level TXEFS[EFFL].

44.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV)

Address offset: 0x0100

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.	PDIV[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PDIV[3:0]**: input clock divider

The APB clock could be divided prior to be used by the CAN sub system. The rate must be computed using the divider output clock.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

44.4.38 FDCAN register map

Table 409. FDCAN register map and reset values[1]

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
0x0000	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	0	1	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	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.
	Reset value																																
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2 [3:0]				DSJW[3:0]			
	Reset value									0	0	0	0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX [1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]						
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS [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	0	0	0	0	0	0	0	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[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
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS [1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030 to 0x003C	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.
	Reset value																																
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[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

Table 409. FDCAN register map and reset values[1] (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
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRSRESI1	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x004C	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.
	Reset value																																
0x0050	FDCAN_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
	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
0x0054	FDCAN_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
	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
0x0058	FDCAN_ILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RxFIFO1	RxFIFO0
	Reset value	0																									0	0	0	0	0	0	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x0060 to 0x007C	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.
	Reset value																																
0x0080	FDCAN_RXGFC	Res.	Res.	Res.	Res.	LSE[3:0].				Res.	Res.	Res.	LSS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
	Reset value					0	0	0	0				0	0	0	0	0							0	0			0	0	0	0	0	0
0x0084	FDCAN_XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	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
0x0088	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLS	Res.	Res.	FIDX[4:0]					MSI [1:0]		Res.	Res.	Res.	BIDX [2:0]		
	Reset value																	0			0	0	0	0	0	0	0				0	0	0
0x0090	FDCAN_RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F0GI [1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
	Reset value							0	0							0	0							0	0					0	0	0	0
0x0094	FDCAN_RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
	Reset value																														0	0	0
0x0098	FDCAN_RXF1S	Res.	Res.	Res.	Res.	Res.	Res.	RF1	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F1GI [1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
	Reset value							0	0							0	0							0	0					0	0	0	0

Table 409. FDCAN register map and reset values[1] (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
0x009C	FDCAN _RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
	Reset value																														0	0	0
0x00A0 to 0x00BC	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.
	Reset value																																
0x00C0	FDCAN _TXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0																							
0x00C4	FDCAN _TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
	Reset value											0				0	0							0	0						0	1	1
0x00C8	FDCAN _TXBRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP2	TRP1	TRP0
	Reset value																														0	0	0
0x00CC	FDCAN _TXBAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR2	AR1	AR0
	Reset value																														0	0	0
0x00D0	FDCAN _TXBCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR2	CR1	CR0
	Reset value																														0	0	0
0x00D4	FDCAN _TXBTO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO2	TO1	TO0
	Reset value																														0	0	0
0x00D8	FDCAN _TXBCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF2	CF1	CF0
	Reset value																														0	0	0
0x00DC	FDCAN _TXBTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE2	TIE1	TIE0
	Reset value																														0	0	0
0x00E0	FDCAN _TXBCIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE2	CFIE1	CFIE0
	Reset value																														0	0	0
0x00E4	FDCAN _TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEF	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EFG [1:0]		Res.	Res.	Res.	Res.	Res.	EFFL [2:0]		
	Reset value							0	0							0	0							0	0						0	0	0
0x00E8	FDCAN _TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI [1:0]	
	Reset value																															0	0
0x0100	FDCAN _CKDIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
	Reset value																													0	0	0	0

1. R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected write, p = Protected set, C = Clear/preset on write.

Refer to *Section 2.2 on page 81* for the register boundary addresses.

45 Universal serial bus full-speed device interface (USB)

45.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus.

USB suspend/resume are supported, which allows to stop the device clocks for low-power consumption.

45.2 USB main features

- USB specification version 2.0 full-speed compliant
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 1024 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support
- Battery Charging Specification Revision 1.2 support
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB_DP line)

45.3 USB implementation

Table 410 describes the USB implementation in the devices.

Table 410. STM32G4 Series USB implementation

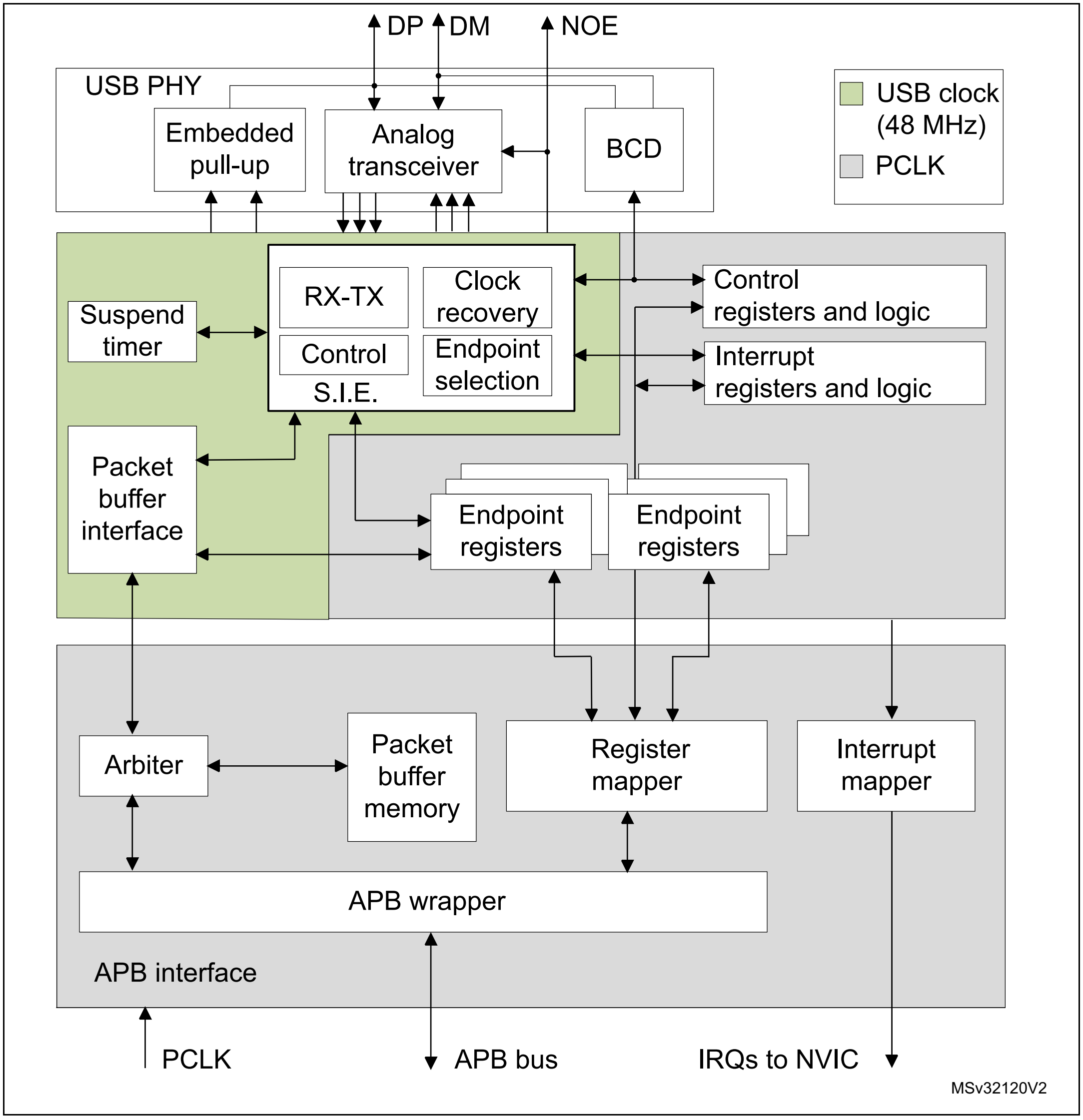
USB features[1]	USB
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	1024 bytes
Dedicated packet buffer memory SRAM access scheme	2 x 16 bits / word
USB 2.0 Link Power Management (LPM) support	X
Battery Charging Detection (BCD) support	X
Embedded pull-up resistor on USB_DP line	X

1. X= supported

45.4 USB functional description

Figure 671 shows the block diagram of the USB peripheral.

Figure 671. USB peripheral block diagram



The USB peripheral provides an USB-compliant connection between the host PC and the function implemented by the microcontroller. Data transfer between the host PC and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 1024 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the USB host, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint is associated with a buffer description block indicating where the endpoint-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint is configured) takes place. The data buffered by the USB peripheral is loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data has been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which allows to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wakeup line to allow the system to immediately restart the normal clock generation and/or support direct clock start/stop.

45.4.1 Description of USB blocks

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB Physical Interface (USB PHY): This block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB_DP line) and support for Battery Charging Detection (BCD), multiplexed on same USB_DP and USB_DM lines. The output enable control signal of the analog transceiver (active low) is provided externally on USB_NOE. It can be used to drive some activity LED or to provide information about the actual communication direction to some other circuitry.
- Serial Interface Engine (SIE): The functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events, such as Start of Frame (SOF), USB_Reset, Data errors etc. and to Endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.
- Timer: This block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- Packet Buffer Interface: This block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the Endpoint registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.

- Endpoint-Related Registers: Each endpoint has an associated register containing the endpoint type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.
- Control Registers: These are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt Registers: These contain the Interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

Note: ** Endpoint 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB1 bus through an APB1 interface, containing the following blocks:

- Packet Memory: This is the local memory that physically contains the Packet Buffers. It can be used by the Packet Buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the Packet Memory is 1024 bytes, structured as 512 half-words of 16 bits.
- Arbiter: This block accepts memory requests coming from the APB1 bus and from the USB interface. It resolves the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.
- Register Mapper: This block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 16-bit wide half-word set addressed by the APB1.
- APB1 Wrapper: This provides an interface to the APB1 for the memory and register. It also maps the whole USB peripheral in the APB1 address space.
- Interrupt Mapper: This block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

45.5 Programming considerations

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

45.5.1 Generic USB device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and Isochronous transfers. Apart from system reset, action is always initiated by the USB peripheral, driven by one of the USB events described below.

45.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software should perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{STARTUP}$ specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the FRES bit in the CNTR register). Clearing the ISTR register then removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB reset (RESET interrupt)

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral will not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the Enable Function (EF) bit of the USB_DADDR register and initializing the EP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RESET interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of reset sequence which triggered the interrupt.

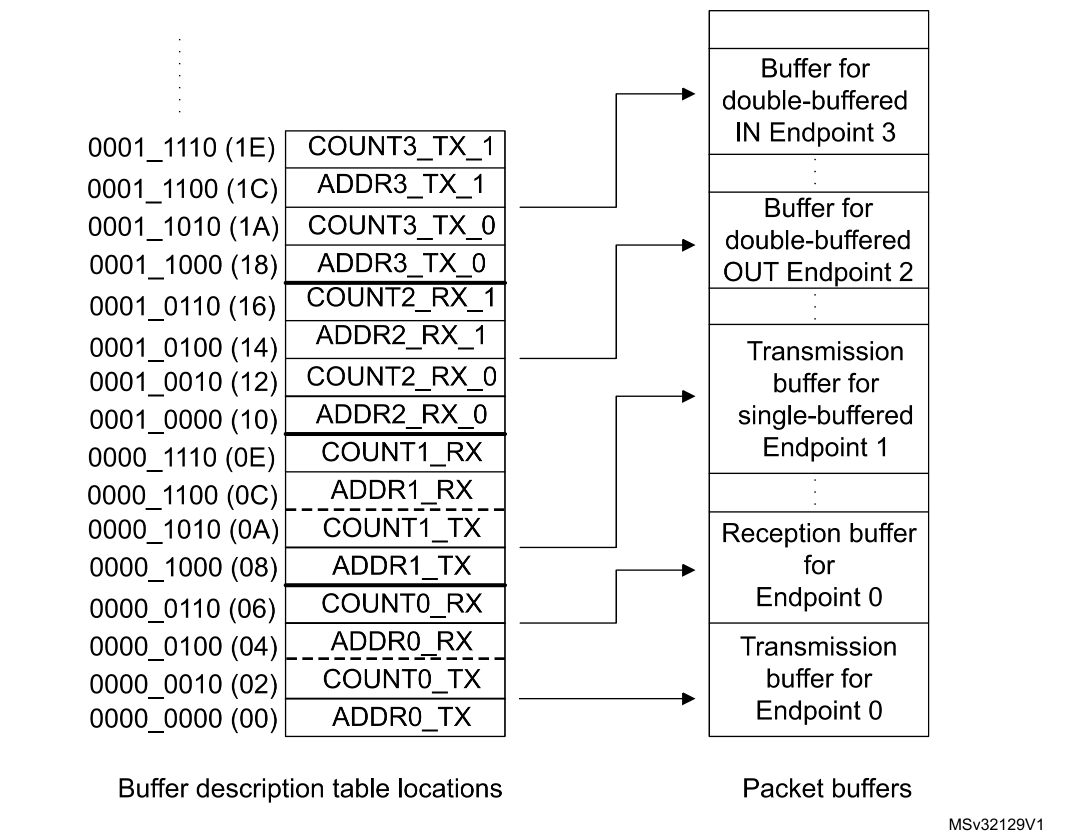
Structure and usage of packet buffers

Each bidirectional endpoint may receive or transmit data from/to the host. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgment. Since the packet buffer memory has to be accessed by the microcontroller also, an arbitration logic takes care of the access conflicts, using half APB1 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both the agents can operate as if the packet memory is a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB1 bus. Different clock configurations are possible where the APB1 clock frequency can be higher or lower than the USB peripheral one.

Note: For USB throughput and system performance reasons during USB active bus connection it is recommended to run APB1 clock no lower than 48 MHz.

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory at the address indicated by the USB_BTABLE register. Each table entry is associated to an endpoint register and it is composed of four 16-bit half-words so that table start address must always be aligned to an 8-byte boundary (the lowest three bits of USB_BTABLE register are always “000”). Buffer descriptor table entries are described in the *Section 45.6.2: Buffer descriptor table*. If an endpoint is unidirectional and it is neither an Isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to *Section 45.5.4: Isochronous transfers* and *Section 45.5.3: Double-buffered endpoints* respectively). The relationship between buffer description table entries and packet buffer areas is depicted in *Figure 672*.

Figure 672. Packet buffer areas with examples of buffer description table locations



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral will never change the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data will be copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX registers so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The EP_TYPE bits in the USB_EPnR register must be set according to the endpoint type, eventually using the EP_KIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STAT_TX bits in the USB_EPnR register and COUNTn_TX must be initialized. For reception, STAT_RX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BL_SIZE and NUM_BLOCK fields. Unidirectional endpoints, except Isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_EPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), should not be modified by the application software, as the hardware can change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

IN packets (data transmission)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of ADDRn_TX and COUNTn_TX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16 bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (Refer to *Structure and usage of packet buffers on page 2012*) and starts sending a DATA0 or DATA1 PID according to USB_EPnR bit DTOG_TX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STAT_TX bits in the USB_EPnR register.

The ADDR internal register is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/2 half-words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed will be used.

On receiving the ACK receipt by the host, the USB_EPnR register is updated in the following way: DTOG_TX bit is toggled, the endpoint is made invalid by setting STAT_TX=10 (NAK) and bit CTR_TX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP_ID and DIR bits in the USB_ISTR register. Servicing of the CTR_TX event starts clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STAT_TX to ‘11 (VALID) to re-enable transmissions. While the STAT_TX bits are equal to ‘10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

OUT and SETUP packets (data reception)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX is stored directly in its internal register ADDR. While COUNT is now reset and the values of BL_SIZE and NUM_BLOCK bit fields, which are read within COUNTn_RX content are used to initialize BUF_COUNT, an internal 16 bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STAT_RX in the USB_EPnR register and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, CRC included (i.e. data payload length + 2), or up to the last allocated memory location, as defined by BL_SIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. in this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer description table entry, leaving unaffected BL_SIZE and NUM_BLOCK fields, which normally do not require to be re-written, and the USB_EPnR register is updated in the following way: DTOG_RX bit is toggled, the endpoint is made invalid by setting STAT_RX = ‘10 (NAK) and bit CTR_RX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP_ID and DIR bits in the USB_ISTR register. The CTR_RX event is serviced by first determining the transaction type (SETUP bit in the USB_EPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being processed. After the received data is processed, the application software should set the STAT_RX bits to ‘11 (Valid) in the USB_EPnR, enabling further transactions. While the STAT_RX bits are equal to ‘10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Control transfers

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOG_TX and DTOG_RX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STAT_TX and STAT_RX are set to ‘10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_EPnR register at each CTR_RX event to distinguish normal OUT transactions from SETUP ones. A USB device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction should be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction should be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software will change NAK to VALID, otherwise to STALL. At the same time, if the status stage will be an OUT, the STATUS_OUT (EP_KIND in the USB_EPnR register) bit should be set, so that an error is generated if a status transaction is performed with not-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STAT_RX to VALID (to accept a new command) and STAT_TX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic doesn’t allow a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STAT_RX bits are set to ‘01 (STALL) or ‘10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued CTR_RX request not yet acknowledged by the application (i.e. CTR_RX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the CTR_RX interrupt.

45.5.3 Double-buffered endpoints

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it will answer with a NAK handshake and the host PC will issue the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_EPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from ‘00 (Disabled): STAT_RX if the double-buffered bulk endpoint is enabled for reception, STAT_TX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_EPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOG_RX (bit 14 of USB_EPnR register) for ‘reception’ double-buffered bulk endpoints or DTOG_TX (bit 6 of USB_EPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral should know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB_EPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_EPnR register bits and DTOG/SW_BUF definition is explained, for the cases of ‘transmission’ and ‘reception’ double-buffered bulk endpoints.

Table 411. Double-buffering buffer flag definition

Buffer flag	‘Transmission’ endpoint	‘Reception’ endpoint
DTOG	DTOG_TX (USB_EPnR bit 6)	DTOG_RX (USB_EPnR bit 14)
SW_BUF	USB_EPnR bit 14	USB_EPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 412. Bulk double-buffering memory buffers usage

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
IN	0	1	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations.
	1	0	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	0	0	None [1]	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	1	1	None [1]	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
OUT	0	1	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.
	1	0	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	0	0	None [1]	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	1	1	None [1]	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.

[1] Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by:

- Writing EP_TYPE bit field at ‘00 in its USB_EPnR register, to define the endpoint as a bulk, and
- Setting EP_KIND bit at ‘1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the CTR_RX or CTR_TX bit of the addressed endpoint USB_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_EPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after

DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains ‘11 (Valid). However, as the token packet of a new transaction is received, the actual endpoint status will be masked as ‘10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see *Table 412 on page 2018*). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing ‘1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control will take place and the actual transfer rate will be limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from ‘11 (Valid) into the STAT bit pair of the related USB_EPnR register. In this case, the USB peripheral will always use the programmed endpoint status, regardless of the buffer usage condition.

45.5.4 Isochronous transfers

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘Isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for Isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, Isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The Isochronous behavior for an endpoint is selected by setting the EP_TYPE bits at ‘10 in its USB_EPnR register; since there is no handshake phase the only legal values for the STAT_RX/STAT_TX bit pairs are ‘00 (Disabled) and ‘11 (Valid), any other value will produce results not compliant to USB standard. Isochronous endpoints implement double-buffering to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOG_RX for ‘reception’ isochronous endpoints, DTOG_TX for ‘transmission’ isochronous endpoints, both in the related USB_EPnR register) according to *Table 413*.

Table 413. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
IN	0	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.
	1	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.
OUT	0	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.
	1	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB_EPnR registers used to implement Isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have Isochronous endpoints enabled both for reception and transmission, two USB_EPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the CTR_RX or CTR_TX bit of the addressed endpoint USB_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_EPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for Isochronous transfers due to the lack of handshake phase, the endpoint remains always ‘11 (Valid). CRC errors or buffer-overrun conditions occurring during Isochronous OUT transfers are anyway considered as correct transactions and they always trigger an CTR_RX event. However, CRC errors will anyway set the ERR bit in the USB_ISTR register to notify the software of the possible data corruption.

45.5.5 Suspend/Resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to ‘1 in USB_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the FSUSP bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set LP_MODE bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to insure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (See “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the LP_MODE bit in USB_CNTR register asynchronously. The resume procedure should be run just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. It is therefore never necessary to handle a WKUP event as an interrupt. If multiple events could have caused the system resume (via EXTI) it is recommended to restart the USB 48 MHz clock as soon as possible. Once this clock is active it is possible to confirm if the event source was the USB resume (i.e. WKUP bit). To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure should address:

1. Optionally turn on external oscillator and/or device PLL, and further wait for USB 48 MHz clock activation (this is required before making any USB register accesses).
2. Clear FSUSP bit of USB_CNTR register, and further wait for USB 48 MHz clock activation (this is required before making any USB register access).
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to *Table 414*, which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the Idle bus state; moreover at the end of a reset sequence the RESET bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which should be handled as usual.

Table 414. Resume event detection

[RXDP,RXDM] status	Wakeup event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode

Table 414. Resume event detection (continued)

[RXDP,RXDM] status	Wakeup event	Required resume software action
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (e.g. a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the RESUME bit in the USB_CNTR register to ‘1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the RESUME bit is clear, the resume sequence will be completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

Note: The RESUME bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the FSUSP bit in USB_CNTR register to 1.

If multiple events could have cause the system resume (via EXTI) it is recommended to re-start the USB 48 MHz clock as soon as possible. Once this clock is active it is possible to confirm if the event source was the USB resume (for instance WKUP bit).

45.6 USB and USB SRAM registers

The USB peripheral registers can be divided into the following groups:

- Common Registers: Interrupt and Control registers
- Endpoint Registers: Endpoint configuration and status

The USB SRAM registers cover:

- Buffer Descriptor Table: Location of packet memory used to locate data buffers (see *Section 2.2: Memory organization* to find USB SRAM base address).

All register addresses are expressed as offsets with respect to the USB peripheral registers base address, except the buffer descriptor table locations, which starts at the USB SRAM base address offset by the value specified in the USB_BTABLE register.

Refer to *Section 1.2 on page 73* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

45.6.1 Common registers

These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.

USB control register (USB_CNTR)

Address offset: 0x40

Reset value: 0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	L1REQM	Res.	L1RESUME	RESUME	FSUSP	LP_MODE	PDWN	FRES
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bit 15 **CTRM:** Correct transfer interrupt mask

0: Correct Transfer (CTR) Interrupt disabled.
1: CTR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 14 **PMAOVRM:** Packet memory area over / underrun interrupt mask

0: PMAOVR Interrupt disabled.
1: PMAOVR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 13 **ERRM:** Error interrupt mask

0: ERR Interrupt disabled.
1: ERR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 12 **WKUPM:** Wakeup interrupt mask

0: WKUP Interrupt disabled.
1: WKUP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 11 **SUSPM:** Suspend mode interrupt mask

0: Suspend Mode Request (SUSP) Interrupt disabled.
1: SUSP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 10 **RESETM:** USB reset interrupt mask

0: RESET Interrupt disabled.
1: RESET Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 9 **SOFM:** Start of frame interrupt mask

0: SOF Interrupt disabled.
1: SOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 8 **ESOFM:** Expected start of frame interrupt mask

0: Expected Start of Frame (ESOF) Interrupt disabled.
1: ESOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 7 **L1REQM:** LPM L1 state request interrupt mask

0: LPM L1 state request (L1REQ) Interrupt disabled.
1: L1REQ Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **L1RESUME:** LPM L1 Resume request

The microcontroller can set this bit to send a LPM L1 Resume signal to the host. After the signaling ends, this bit is cleared by hardware.

Bit 4 **RESUME:** Resume request

The microcontroller can set this bit to send a Resume signal to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the Host PC is ready to drive the resume sequence up to its end.

Bit 3 **FSUSP:** Force suspend

Software must set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms.
0: No effect.
1: Enter suspend mode. Clocks and static power dissipation in the analog transceiver are left unaffected. If suspend power consumption is a requirement (bus-powered device), the application software should set the LP_MODE bit after FSUSP as explained below.

Bit 2 **LP_MODE:** Low-power mode

This mode is used when the suspend-mode power constraints require that all static power dissipation is avoided, except the one required to supply the external pull-up resistor. This condition should be entered when the application is ready to stop all system clocks, or reduce their frequency in order to meet the power consumption requirements of the USB suspend condition. The USB activity during the suspend mode (WKUP event) asynchronously resets this bit (it can also be reset by software).
0: No Low-power mode.
1: Enter Low-power mode.

Bit 1 **PDWN:** Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.
0: Exit Power Down.
1: Enter Power down mode.

Bit 0 **FRES:** Force USB Reset

0: Clear USB reset.
1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.

USB interrupt status register (USB_ISTR)

Address offset: 0x44

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	L1REQ	Res.	Res.	DIR	EP_ID[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r	r	r	r

This register contains the status of all the interrupt sources allowing application software to determine, which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, will perform all necessary actions, and finally it will clear the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line will be kept high again. If several bits are set simultaneously, only a single interrupt will be generated.

Endpoint transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_EPnR register (the CTR bit is actually a read only bit). For endpoint-related interrupts, the software can use the Direction of Transaction (DIR) and EP_ID read-only bits to identify, which endpoint made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt will be requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with ‘0 (these bits can only be cleared by software). Read-modify-write cycles should be avoided because between the read and the write operations another bit could be set by the hardware and the next write will clear it before the microprocessor has the time to serve the event.

The following describes each bit in detail:

Bit 15 **CTR:** Correct transfer

This bit is set by the hardware to indicate that an endpoint has successfully completed a transaction; using DIR and EP_ID bits software can determine which endpoint requested the interrupt. This bit is read-only.

Bit 14 **PMAOVR:** Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host will retry the transaction. The PMAOVR interrupt should never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during Isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 13 **ERR:** Error

This flag is set whenever one of the errors listed below has occurred:
NANS: No ANSwer. The timeout for a host response has expired.
CRC: Cyclic Redundancy Check error. One of the received CRCs, either in the token or in the data, was wrong.
BST: Bit Stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.
FVIO: Framing format Violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).
The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (e.g. loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 12 **WKUP:** Wakeup

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the LP_MODE bit in the CTLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (e.g. wakeup unit) about the start of the resume process. This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 11 **SUSP:** Suspend mode request

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (FSUSP=1) until the end of resume sequence. This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 10 **RESET:** USB reset request

Set when the USB peripheral detects an active USB RESET signal at its inputs. The USB peripheral, in response to a RESET, just resets its internal protocol state machine, generating an interrupt if RESETM enable bit in the USB_CNTR register is set. Reception and transmission are disabled until the RESET bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RESET interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.

This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 9 **SOF:** Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine may monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this could be useful for isochronous applications).

This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 8 **ESOF:** Expected start of frame

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the Suspend Timer issues this interrupt. If three consecutive ESOF interrupts are generated (i.e. three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the Suspend Timer is not yet locked. This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 7 **L1REQ:** LPM L1 state request

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **DIR:** Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.

If DIR bit=0, CTR_TX bit is set in the USB_EPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).

If DIR bit=1, CTR_RX bit or both CTR_TX/CTR_RX are set in the USB_EPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed.

This information can be used by the application software to access the USB_EPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **EP_ID[3:0]:** Endpoint Identifier

These bits are written by the hardware according to the endpoint number, which generated the interrupt request. If several endpoint transactions are pending, the hardware writes the endpoint identifier related to the endpoint having the highest priority defined in the following way: Two endpoint sets are defined, in order of priority: Isochronous and double-buffered bulk endpoints are considered first and then the other endpoints are examined. If more than one endpoint from the same set is requesting an interrupt, the EP_ID bits in USB_ISTR register are assigned according to the lowest requesting endpoint register, EP0R having the highest priority followed by EP1R and so on. The application software can assign a register to each endpoint according to this priority scheme, so as to order the concurring endpoint requests in a suitable way. These bits are read only.

USB frame number register (USB_FNR)

Address offset: 0x48

Reset value: 0x0XXX where X is undefined

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 15 **RXDP:** Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wakeup event.

Bit 14 **RXDM:** Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wakeup event.

Bit 13 **LCK:** Locked

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]:** Lost SOF

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]:** Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for Isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

USB device address (USB_DADDR)

Address offset: 0x4C

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.	EF	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved

Bit 7 **EF:** Enable function

This bit is set by the software to enable the USB device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at ‘0 no transactions are handled, irrespective of the settings of USB_EPnR registers.

Bits 6:0 **ADD[6:0]:** Device address

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the Endpoint Address (EA) field in the associated USB_EPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

Buffer table address (USB_BTABLE)

Address offset: 0x50

Reset value: 0x0000

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

Bits 15:3 **BTABLE[15:3]:** Buffer table

These bits contain the start address of the buffer allocation table inside the dedicated packet memory. This table describes each endpoint buffer location and size and it must be aligned to an 8 byte boundary (the 3 least significant bits are always ‘0). At the beginning of every transaction addressed to this device, the USB peripheral reads the element of this table related to the addressed endpoint, to get its buffer start location and the buffer size (Refer to *Structure and usage of packet buffers on page 2012*).

Bits 2:0 Reserved, forced by hardware to 0.

LPM control and status register (USB_LPMCSR)

Address offset: 0x54

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.	BESL[3:0]				REM WAKE	Res.	LPM ACK	LPM EN
								r	r	r	r	r		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:4 **BESL[3:0]:** BESL value

These bits contain the BESL value received with last ACKed LPM Token

Bit 3 **REMWAKE:** bRemoteWake value

This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved

Bit 1 **LPMACK:** LPM Token acknowledge enable

0: the valid LPM Token will be NYET.
1: the valid LPM Token will be ACK.
The NYET/ACK will be returned only on a successful LPM transaction:
No errors in both the EXT token and the LPM token (else ERROR)
A valid bLinkState = 0001B (L1) is received (else STALL)

Bit 0 **LPMEN:** LPM support enable

This bit is set by the software to enable the LPM support within the USB device. If this bit is at ‘0 no LPM transactions are handled.

Battery charging detector (USB_BCDR)

Address offset: 0x58

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPPU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2 DET	SDET	PDET	DC DET	SDEN	PDEN	DCD EN	BCD EN
rw								r	r	r	r	rw	rw	rw	rw

Bit 15 **DPPU:** DP pull-up control

This bit is set by software to enable the embedded pull-up on the DP line. Clearing it to ‘0’ can be used to signalize disconnect to the host when needed by the user software.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET:** DM pull-up detection status

This bit is active only during PD and gives the result of comparison between DM voltage level and V_{LGC} threshold. In normal situation, the DM level should be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, ACA, CDP or DCP).
1: PS2 port or proprietary charger detected.

Bit 6 **SDET:** Secondary detection (SD) status

This bit gives the result of SD.
0: CDP detected.
1: DCP detected.

Bit 5 **PDET:** Primary detection (PD) status

This bit gives the result of PD.
0: no BCD support detected (connected to SDP or proprietary device).
1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 **DCDET:** Data contact detection (DCD) status

This bit gives the result of DCD.
0: data lines contact not detected.
1: data lines contact detected.

Bit 3 **SDEN:** Secondary detection (SD) mode enable

This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 2 **PDEN:** Primary detection (PD) mode enable

This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 1 **DCDEN:** Data contact detection (DCD) mode enable

This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 0 **BCDEN:** Battery charging detector (BCD) enable

This bit is set by the software to enable the BCD support within the USB device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD should be placed in OFF mode by clearing this bit to ‘0 in order to allow the normal USB operation.

Endpoint-specific registers

The number of these registers varies according to the number of endpoints that the USB peripheral is designed to handle. The USB peripheral supports up to 8 bidirectional endpoints. Each USB device must support a control endpoint whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint number value. For each endpoint, an USB_EPnR register is available to store the endpoint specific information.

USB endpoint n register (USB_EPnR), n=[0..7]

Address offset: 0x00 to 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_ RX	DTOG _RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_ KIND	CTR_ TX	DTOG_ TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

They are also reset when an USB reset is received from the USB bus or forced through bit FRES in the CTLR register, except the CTR_RX and CTR_TX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint has its USB_EPnR register where *n* is the endpoint identifier.

Read-modify-write cycles on these registers should be avoided because between the read and the write operations some bits could be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.

Bit 15 **CTR_RX:** Correct transfer for reception

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only ‘0 can be written, writing 1 has no effect.

Bit 14 **DTOG_RX:** Data toggle, for reception transfers

If the endpoint is not Isochronous, this bit contains the expected value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent to the USB host, following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to *Section 45.5.3: Double-buffered endpoints*).

If the endpoint is Isochronous, this bit is used only to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to *Section 45.5.4: Isochronous transfers*). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes ‘0, the value of DTOG_RX remains unchanged, while writing ‘1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STAT_RX [1:0]:** Status bits, for reception transfers

These bits contain information about the endpoint status, which are listed in *Table 415: Reception status encoding on page 2034*.These bits can be toggled by software to initialize their value. When the application software writes ‘0, the value remains unchanged, while writing ‘1 makes the bit value toggle. Hardware sets the STAT_RX bits to NAK when a correct transfer has occurred (CTR_RX=1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledge a new transaction

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to *Section 45.5.3: Double-buffered endpoints*).

If the endpoint is defined as Isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT_RX bits to ‘STALL’ or ‘NAK’ for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing ‘1.

Bit 11 **SETUP:** Setup transaction completed

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (CTR_RX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while CTR_RX bit is at 1; its state changes when CTR_RX is at 0. This bit is read-only.

Bits 10:9 **EP_TYPE[1:0]:** Endpoint type

These bits configure the behavior of this endpoint as described in *Table 416: Endpoint type encoding on page 2034*. Endpoint 0 must always be a control endpoint and each USB function must have at least one control endpoint which has address 0, but there may be other control endpoints if required. Only control endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint is defined as NAK, the USB peripheral will not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint is defined as STALL in the receive direction, then the SETUP packet will be accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint is a control one.

Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EP_KIND configuration bit.

The usage of Isochronous endpoints is explained in *Section 45.5.4: Isochronous transfers*

Bit 8 **EP_KIND:** Endpoint kind

The meaning of this bit depends on the endpoint type configured by the EP_TYPE bits. *Table 417* summarizes the different meanings.

DBL_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in *Section 45.5.3: Double-buffered endpoints*.

STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered ‘STALL’ instead of ‘ACK’. This bit may be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **CTR_TX:** Correct Transfer for transmission

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only ‘0 can be written.

Bit 6 **DTOG_TX:** Data Toggle, for transmission transfers

If the endpoint is non-isochronous, this bit contains the required value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to *Section 45.5.3: Double-buffered endpoints*)

If the endpoint is Isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to *Section 45.5.4: Isochronous transfers*). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for Isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes ‘0, the value of DTOG_TX remains unchanged, while writing ‘1 makes the bit value toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STAT_TX [1:0]:** Status bits, for transmission transfers

These bits contain the information about the endpoint status, listed in *Table 418*. These bits can be toggled by the software to initialize their value. When the application software writes ‘0, the value remains unchanged, while writing ‘1 makes the bit value toggle. Hardware sets the STAT_TX bits to NAK, when a correct transfer has occurred (CTR_TX=1) corresponding to a IN or SETUP (control only) transaction addressed to this endpoint. It then waits for the software to prepare the next set of data to be transmitted.

Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to *Section 45.5.3: Double-buffered endpoints*).

If the endpoint is defined as Isochronous, its status can only be “VALID” or “DISABLED”. Therefore, the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT_TX bits to ‘STALL’ or ‘NAK’ for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing ‘1.

Bits 3:0 **EA[3:0]:** Endpoint address

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

Table 415. Reception status encoding

STAT_RX[1:0]	Meaning
00	**DISABLED:** all reception requests addressed to this endpoint are ignored.
01	**STALL**: the endpoint is stalled and all reception requests result in a STALL handshake.
10	**NAK**: the endpoint is naked and all reception requests result in a NAK handshake.
11	**VALID**: this endpoint is enabled for reception.

Table 416. Endpoint type encoding

EP_TYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 417. Endpoint kind meaning

EP_TYPE[1:0]		EP_KIND meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	Not used
11	INTERRUPT	Not used

Table 418. Transmission status encoding

STAT_TX[1:0]	Meaning
00	**DISABLED:** all transmission requests addressed to this endpoint are ignored.
01	**STALL**: the endpoint is stalled and all transmission requests result in a STALL handshake.
10	**NAK**: the endpoint is naked and all transmission requests result in a NAK handshake.
11	**VALID**: this endpoint is enabled for transmission.

45.6.2 Buffer descriptor table

Note: *The buffer descriptor table is located inside the packet buffer memory in the separate "USB SRAM" address space.*

Although the buffer descriptor table is located inside the packet buffer memory ("USB SRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USB SRAM base address. The buffer descriptor table entry associated with the USB_EPnR registers is described below. The packet memory should be accessed only by byte (8-bit) or half-word (16-bit) accesses. Word (32-bit) accesses are not allowed.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in *Structure and usage of packet buffers on page 2012*.

Transmission buffer address n (USB_ADDRn_TX)

Address offset: [USB_BTABLE] + n*8

Note: *In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB_ADDRn_TX_0.*

In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB_ADDRn_RX_0.

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

Bits 15:1 **ADDRn_TX[15:1]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint associated with the USB_EPnR register at the next IN token addressed to it.

Bit 0 Must always be written as ‘0 since packet memory is half-word wide and all packet buffers must be half-word aligned.

Transmission byte count n (USB_COUNTn_TX)

Address offset: [USB_BTABLE] + n*8 + 2

Note: *In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB_COUNTn_TX_0.*

In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB_COUNTn_RX_0.

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

Bits 15:10 These bits are not used since packet size is limited by USB specifications to 1023 bytes. Their value is not considered by the USB peripheral.

Bits 9:0 **COUNTn_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint associated with the USB_EPnR register at the next IN token addressed to it.

Reception buffer address n (USB_ADDRn_RX)

Address offset: [USB_BTABLE] + n*8 + 4

Note: In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB_ADDRn_RX_1.

In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB_ADDRn_TX_1.

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

Bits 15:1 **ADDRn_RX[15:1]:** Reception buffer address

These bits point to the starting address of the packet buffer, which will contain the data received by the endpoint associated with the USB_EPnR register at the next OUT/SETUP token addressed to it.

Bit 0 This bit must always be written as ‘0 since packet memory is half-word wide and all packet buffers must be half-word aligned.

Reception byte count n (USB_COUNTn_RX)

Address offset: [USB_BTABLE] + n*8 + 6

Note: In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB_COUNTn_RX_1.

In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB_COUNTn_TX_1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLSIZE	NUM_BLOCK[4:0]					COUNTn_RX[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (See “Universal Serial Bus Specification”).

Bit 15 **BL_SIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BL_SIZE=0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BL_SIZE=1, the memory block is 32-byte large, which allows to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 14:10 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BL_SIZE value as illustrated in *Table 419*.

Bits 9:0 **COUNTn_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint associated with the USB_EPnR register during the last OUT/SETUP transaction addressed to it.

Table 419. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BL_SIZE=0	Memory allocated when BL_SIZE=1
0 (‘00000)	Not allowed	32 bytes
1 (‘00001)	2 bytes	64 bytes
2 (‘00010)	4 bytes	96 bytes
3 (‘00011)	6 bytes	128 bytes
...	...	...
14 (‘01110)	28 bytes	480 bytes
15 (‘01111)	30 bytes	
16 (‘10000)	32 bytes	
...	...	...
29 (‘11101)	58 bytes	
30 (‘11110)	60 bytes	
31 (‘11111)	62 bytes	N/A

45.6.3 USB register map

The table below provides the USB register map and reset values.

Table 420. USB 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	USB_EP0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USB_EP1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USB_EP2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USB_EP3R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USB_EP4R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USB_EP5R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USB_EP6R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	USB_EP7R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x3F	Reserved																																
0x40	USB_CNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	L1REQM	Res.	L1RESUME	RESUME	FSUSP	LP_MODE	PDWN	FRES
	Reset value																	0	0	0	0	0	0	0	0	0		0	0	0	0	1	1
0x44	USB_ISTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	L1REQ	Res.	Res.	DIR	EP_ID[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x48	USB_FNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	USB_DADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0

Table 420. USB 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
0x50	**USB_BTABLE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BTABLE[15:3]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x54	**USB_LPMCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REMWAKE	Res.	LPMACK	LPMEN
	Reset value																									0	0	0	0	0		0	0
0x58	**USB_BCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPPU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET	SDEN	PDEN	DCDEN	BCDEN
	Reset value																	0								0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 81* for the register boundary addresses.

46 USB Type-C™ / USB Power Delivery interface (UCPD)

46.1 Introduction

The USB Type-C / USB Power Delivery interface complies with:

- Universal Serial Bus Type-C Cable and Connector Specification: release 2.0, August, 2019
- Universal Serial Bus Power Delivery specifications:
  - revision 2.0, version 1.3, January 12, 2017
  - revision 3.0, version 2.0, August 29, 2019

It integrates the physical layer of the Power Delivery (PD) specification, with CC signaling method (no VBUS), for operation with Type-C cables.

46.2 UCPD main features

- Compliance with USB Type-C specification release 2.0
- Compliance with USB Power Delivery specifications revision 2.0 and 3.0
  - Enabling advanced applications such as PPS (programmable power supply)
- Stop mode low-power operation support
- Built-in analog PHY
  - USB Type-C pull-up (Rp, all values) and pull-down (Rd) resistors
  - “Dead battery” Rd support
  - USB Power Delivery message transmission and reception
  - FRS (fast role swap) Rx support
- Digital controller
  - BMC (bi-phase mark coding) encode and decode
  - 4b5b encode and decode
  - USB Type-C level detection with de-bounce, generating interrupts
  - FRS detection, generating an interrupt
  - DMA-compatible byte-level interface for USB Power Delivery payload, generating interrupts
  - USB Power Delivery clock pre-scaler / dividers
  - CRC generation/checking
  - Support of ordered sets, with a programmable ordered set mask at receive
  - Clock recovery from incoming Rx stream

46.3 UCPD implementation

The devices have one UCPD controller to support one USB Type-C port.

Table 421. UCPD implementation[1]

UCPD feature	UCPD1
Dead battery support via UCPDx_DBCC1 and UCPDx_DBCC2 external signals	X
UCPDx_FRSTX as alternate function pin	X
Fully automatic trimming (no SW override necessary)	X
Discrete component PHY support	-

[1] “X” = supported, “-” = not supported

46.4 UCPD functional description

The UCPD peripheral provides hardware support for the USB Power Delivery control interface specification, using I/Os specifically designed for that purpose.

The built-in PHY directly detects Type-C voltage levels, supports Power Delivery BIST carrier mode 2 (Tx only), BIST test data (Tx and Rx), and Power Delivery Rx FRS signaling.

For Power Delivery FRS Tx signaling, the device can be configured to control, through UCPD_FRSTX pin (alternate function), external NMOS transistors that ensure low-resistance pull-down on CC lines.

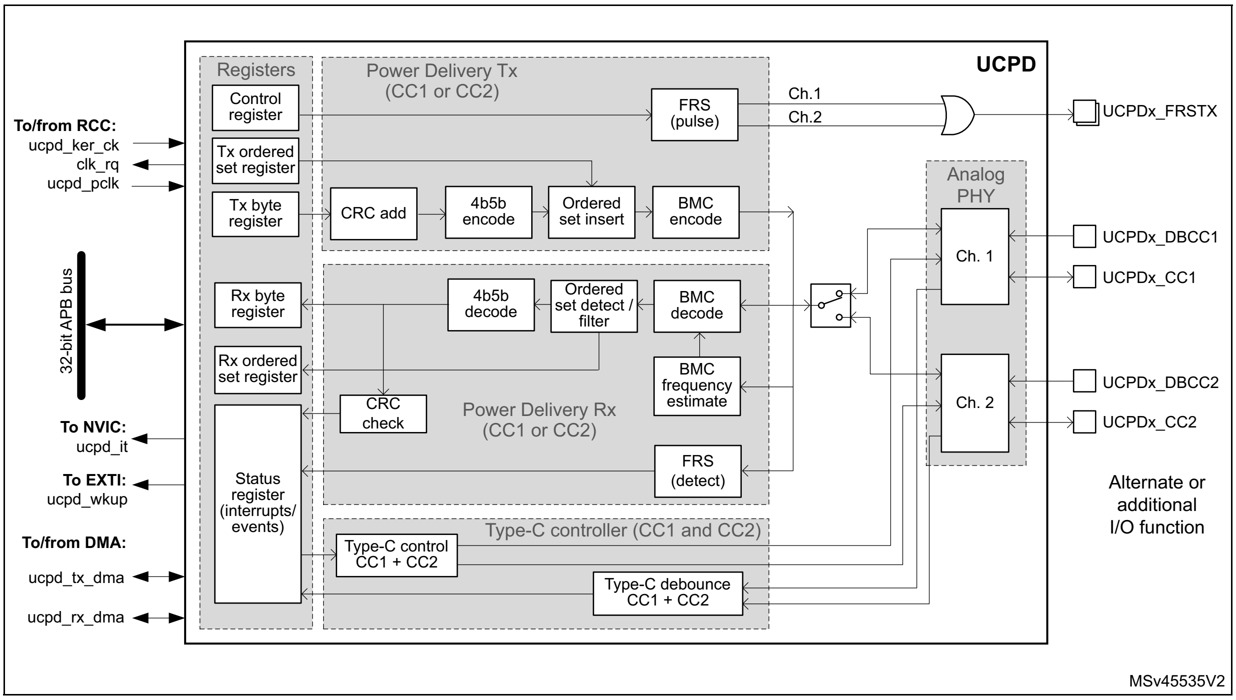
The UCPD transmitter BMC (bi-phase mark) encodes and transmits data: preamble, SOP, payload data from protocol layer (after 4b5b-encoding), CRC, and EOP on the Type-C connector CC lines. It automatically inserts inter-frame gap and executes “Hard Reset”.

The UCPD receiver detects SOP, BMC-decodes the incoming stream, recovers the preamble, 4b5b-decodes payload data, detects EOP, and checks CRC. It automatically detects five K-code SOP and two Reset ordered sets, plus two software-defined patterns (allows for only three out of four K-codes being correctly received, as defined by the standard).

In Stop mode, the peripheral maintains the ability to detect incoming USB Power Delivery messages and FRS signaling, which allows low-power operation.

46.4.1 UCPD block diagram

Figure 673. UCPD block diagram



The following table lists external signals (alternate or additional I/O functions).

Table 422. UCPD signals on pins

Pin name	Signal type	Description
UCPDx_FRSTX	Output	USB Type-C fast role swap (FRS) signaling control, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the active CC line. A typical application has two such transistors (one per CC line) and reserves a separate I/O to drive either NMOS. Initially, the I/Os are configured as low-driving GPIOs. Upon detecting, through the Type-C state machine, the orientation of the cable attached, which determines the active CC line, the I/O of the active CC line must be set to its UCPDx_FRSTX alternate function and the I/O of the inactive CC line as low-driving GPIO.
UCPDx_CC1	Input/output	USB Type-C configuration control line 1, to be routed to the USB Type-C connector CC1 terminal.
UCPDx_CC2	Input/output	USB Type-C configuration control line 2, to be routed to the USB Type-C connector CC2 terminal.
UCPDx_DBCC1	Input	USB Type-C configuration control line 1 dead battery signal, to be routed to the USB Type-C connector CC1 terminal if dead battery support is required.
UCPDx_DBCC2	Input	USB Type-C configuration control line 2 dead battery signal, to be routed to the USB Type-C connector CC2 terminal if dead battery support is required.

The following table lists key internal signals.

Table 423. UCPD internal signals

Internal signal name	Signal type	Description
ucpd_pclk	Input	APB clock for registers
ucpd_ker_ck	Input	Kernel clock
ucpd_tx_dma	Input/Output	Rx DMA acknowledge / request
ucpd_rx_dma	Input/Output	Tx DMA acknowledge / request
ucpd_it	Output	Interrupt request (all interrupts OR-ed) connected to NVIC
ucpd_wkup	Output	Wakeup request connected to EXTI
clk_rq	Output	Clock request connected to RCC

46.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

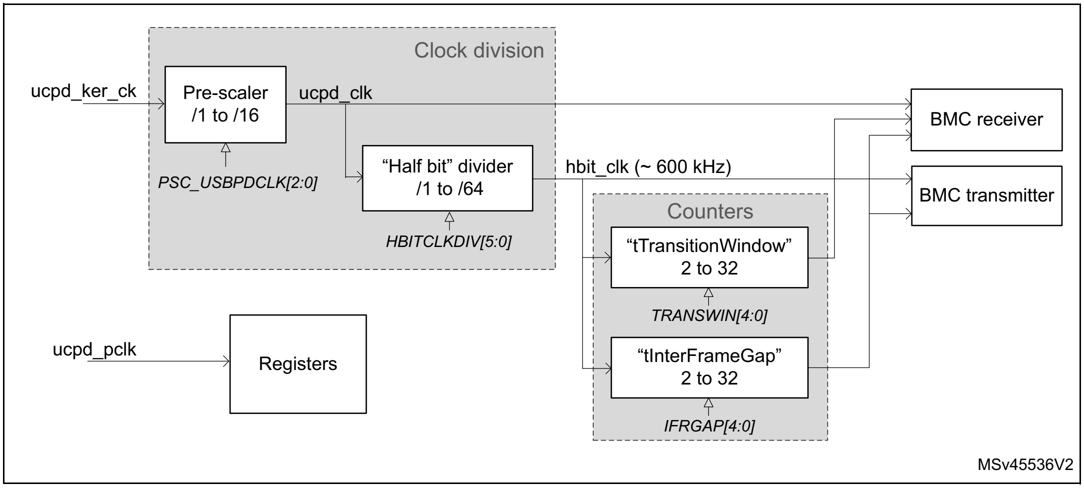
The register section is clocked with the APB clock (ucpd_pclk).

The main functional part of the transmitter is clocked with ucpd_clk clock, pre-scaled from the ucpd_ker_ck (HSI16) clock according to the PSC_USBPDCLK[2:0] bitfield of the UCPD_CFGR1 register. The main functional part of the receiver is clocked with the ucpd_rx_clk recovered from the incoming bitstream.

The receiver is designed to work in the clock frequency range from 6 to 18 MHz. However, the optimum performance is ensured in the range from 9 to 18 MHz.

The following diagram shows the clocking and timing elements of the UCPD peripheral.

Figure 674. Clock division and timing elements



Refer to the USB PD specification in order to set appropriate delays. For *tTransitionWindow* and especially for *tInterFrameGap*, the clock frequency uncertainty must be taken into account so as to respect specified timings in all cases.

46.4.3 Physical layer protocol

The physical layer covers the signaling underlying the USB Power Delivery specification.

On the transmitter side its main function is to form packets according to the defined packet format including generally:

- preamble
- start of packet (SOP, ordered set)
- payload header
- payload data
- cyclic redundancy check (CRC) information
- end of packet (EOP)

Before going on the CC line, the data stream is BMC-encoded, respecting specified timing restrictions.

On the receive side, the principle task is to:

- extract start of packet (SOP, ordered set) information
- extract payload header
- extract payload data
- receive and check CRC
- receive end of packet (EOP)

The receive is basically a reverse of the transmit process, thus starting with BMC data stream decoding.

Symbol encoding

Apart from the preamble all symbols are encoded with a 4b5b scheme according to the specification shown in the following table.

Table 424. 4b5b Symbol Encoding Table

Name	4b	5b	Symbol description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C

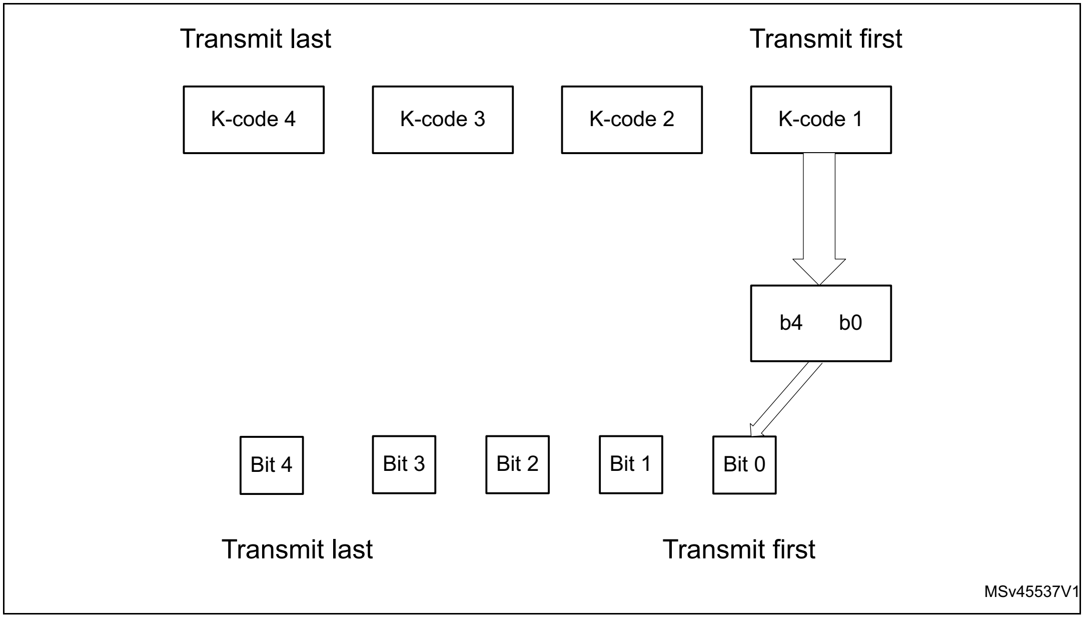
Table 424. 4b5b Symbol Encoding Table (continued)

Name	4b	5b	Symbol description
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
Sync-1	K-code	11000	Startsynch #1
Sync-2	K-code	10001	Startsynch #2
RST-1	K-code	00111	Hard Reset #1
RST-2	K-code	11001	Hard Reset #2
EOP	K-code	01101	EOP
Reserved	Error	00000	Do Not Use
Reserved	Error	00001	Do Not Use
Reserved	Error	00010	Do Not Use
Reserved	Error	00011	Do Not Use
Reserved	Error	00100	Do Not Use
Reserved	Error	00101	Do Not Use
Sync-3	K-code	00110	Startsynch #3
Reserved	Error	01000	Do Not Use
Reserved	Error	01100	Do Not Use
Reserved	Error	10000	Do Not Use
Reserved	Error	11111	Do Not Use

Ordered sets

An ordered set consists of four K-codes as shown in the following figure.

Figure 675. K-code transmission



The following table lists the defined ordered sets, including all possible SOP* sequences.

At the physical layer, the Hard Reset has higher priority than the other ordered sets so it can interrupt an on-going Tx message.

Table 425. Ordered sets

Ordered set name	K-code #1	K-code #2	K-code #3	K-code #4
SOP	Sync-1	Sync-1	Sync-1	Sync-2
SOP’	Sync-1	Sync-1	Sync-3	Sync-3
SOP”	Sync-1	Sync-3	Sync-1	Sync-3
Hard Reset	RST-1	RST-1	RST-1	RST-2
Cable Reset	RST-1	Sync-1	RST-1	Sync-3
SOP’_Debug	Sync-1	RST-2	RST-2	Sync-3
SOP”_Debug	Sync-1	RST-2	Sync-3	Sync-2

On reception, the physical layer must accept ordered sets with any combination of three correct K-codes out of four, as shown in the following table:

Table 426. Validation of ordered sets

Status	1st code	2nd code	3rd code	4th code
Valid	Corrupt	K-code	K-code	K-code
Valid	K-code	Corrupt	K-code	K-code

Table 426. Validation of ordered sets (continued)

Status	1st code	2nd code	3rd code	4th code
Valid	K-code	K-code	Corrupt	K-code
Valid	K-code	K-code	K-code	Corrupt
Valid (perfect)	K-code	K-code	K-code	K-code
Not valid (example)	K-code	Corrupt	K-code	Corrupt

Bit ordering at transmission

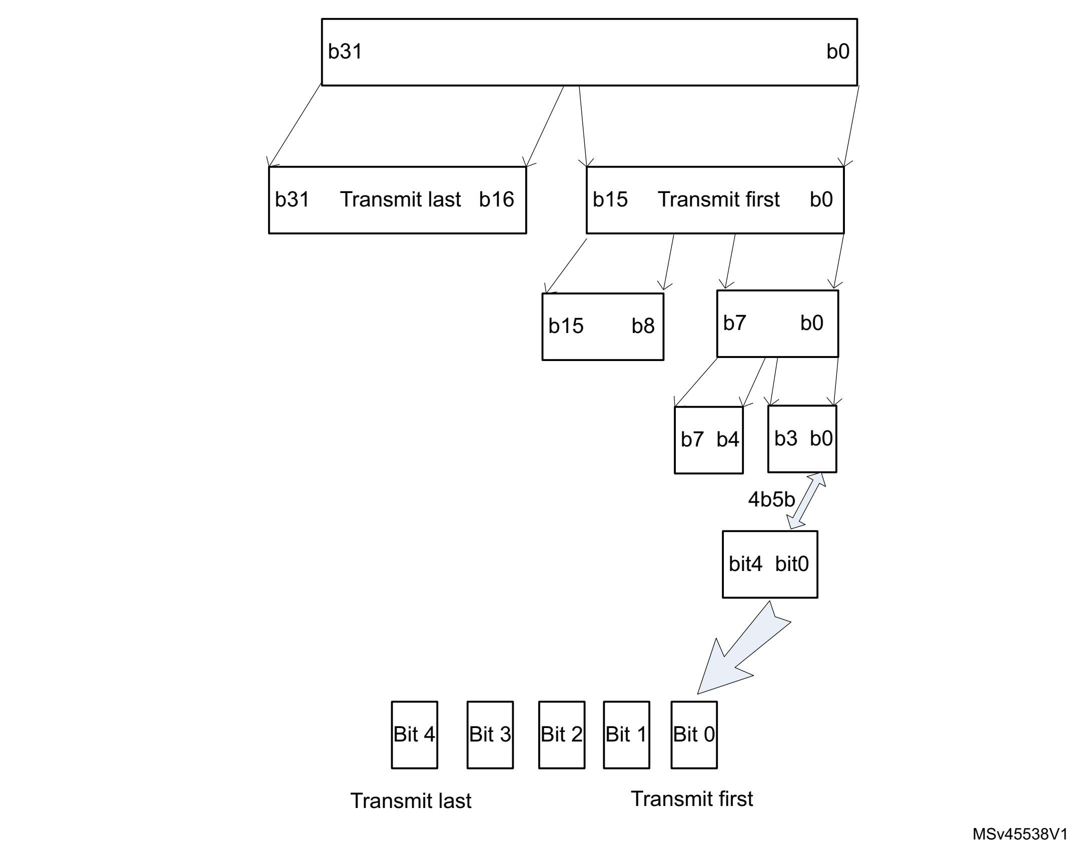
Allowed transmission data units / data sizes are in the following table.

Table 427. Data size

Data unit	Non-encoded	Encoded
Byte	8-bits	10-bits
Word	16-bits	20-bits
DWord	32-bits	40-bits

The bit transmission order is shown in the following figure.

Figure 676. Transmit order for various sizes of data

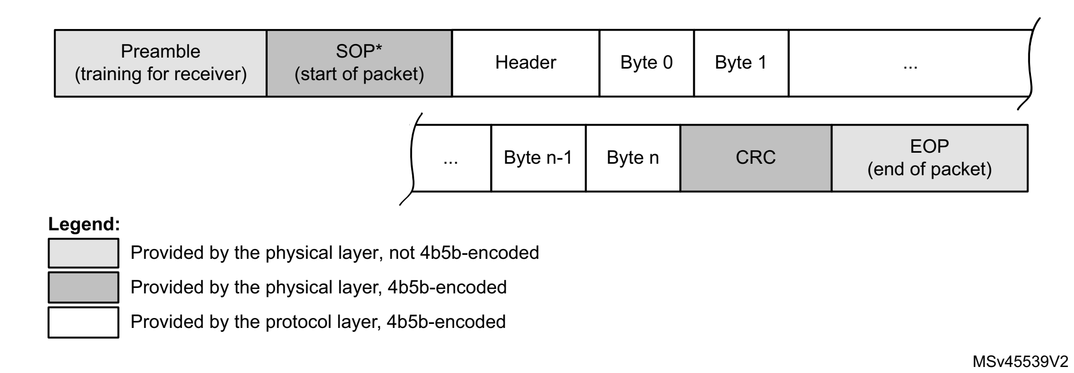


Packet format

Messages other than Hard Reset and Cable Reset

The packet format is shown in the following figure, with information on 4b5b encode and data source.

Figure 677. Packet format


Preamble (training for receiver)
SOP* (start of packet)
Header
Byte 0
Byte 1
...
...
Byte n-1
Byte n
CRC
EOP (end of packet)
Legend:
Provided by the physical layer, not 4b5b-encoded
Provided by the physical layer, 4b5b-encoded
Provided by the protocol layer, 4b5b-encoded
MSv45539V2

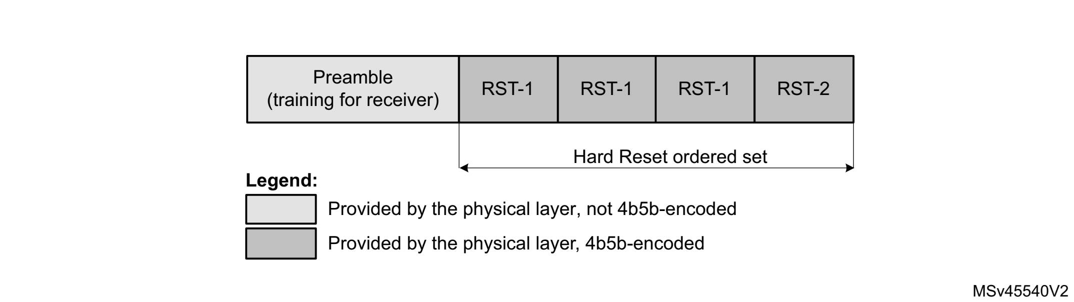
Hard Reset

The physical layer handles the Hard Reset signaling differently than the other types of message as it has higher priority to be able to interrupt an on-going transfer.

The physical layer specification implies the following sequence in the case of an ongoing Tx message:

1. Terminate the message by sending an EOP K-code and discard the rest of the message.
2. Wait for *tInterFrameGap* time.
3. If the CC line is not idle, wait until it goes idle.
4. Send the preamble followed by the four K-codes of Hard Reset signaling.
5. Disable the CC channel (stop sending and receiving), reset the physical layer and inform the protocol layer that the physical layer is reset.
6. Re-enable the channel when requested by the protocol layer.

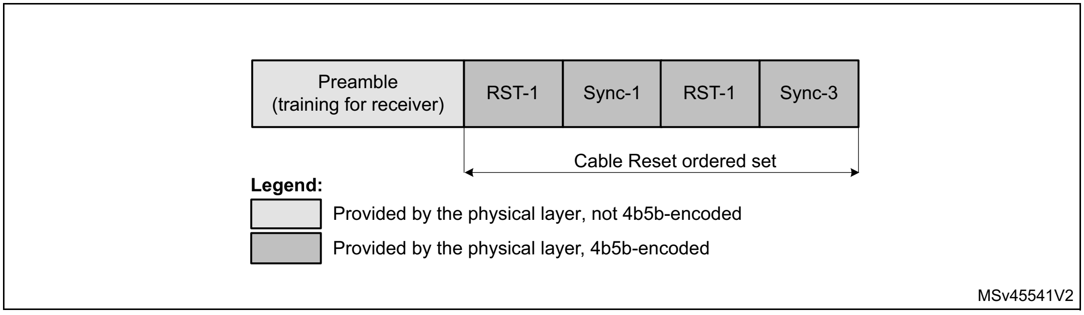
Figure 678. Line format of Hard Reset


Preamble (training for receiver)
RST-1
RST-1
RST-1
RST-2
Hard Reset ordered set
Legend:
Provided by the physical layer, not 4b5b-encoded
Provided by the physical layer, 4b5b-encoded
MSv45540V2

Cable Reset

Cable Reset shown in the following figure is similar in format to Hard Reset, but unlike Hard Reset it does not require a specific high-priority treatment.

Figure 679. Line format of Cable Reset



Collision avoidance

The physical layer respects the *tInterFrameGap* delay between end of last-transmitted bit of a Tx message, and the first bit of a following message.

It also checks the idle state of the CC line before starting transmission. The CC line is considered idle if it shows less than three (*nTransitionCount*) transitions within *tTransitionWindow* (12 to 20 µs). The Power Delivery specification revision 3.0 also requires to manage the Rd value (source) and monitor Type-C voltage level for these Rp modifications (at the sink).

Physical layer signaling schemes

The bit are signaled with bi-phase mark coding (BMC).

BIST

Depending on the BIST action required by the protocol layer, either of the following can be run:

- a Tx BIST pattern test, achieved by writing TXMODE and TXSEND
- an Rx BIST pattern test, achieved by writing RXMODE to the correct value for RXBIST.

The two possible patterns supported in UCPD (corresponding to “BMC” mode) are:

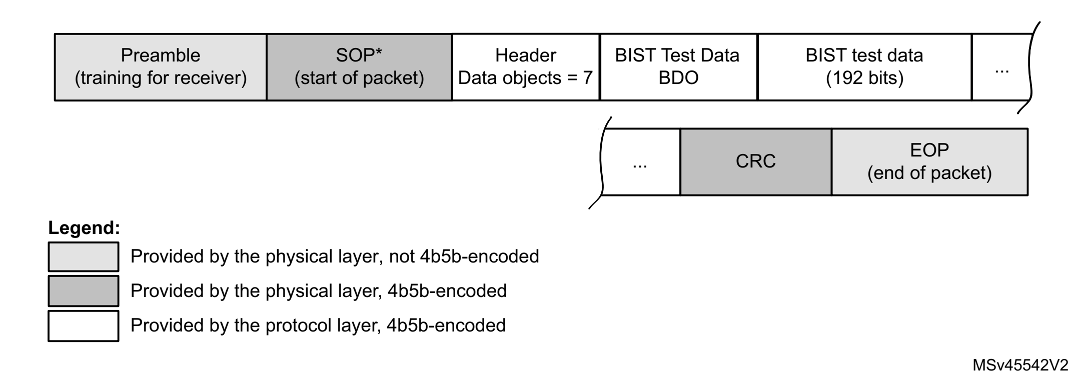
- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), applies to Tx only. As opposed to Tx, the receiver in this mode simply ignores the CC line during this state.

BIST test data pattern

The test data pattern is not viewed as a special case in UCPD.

The BIST test data packet frame format is shown in the following figure.

Figure 680. BIST test data frame



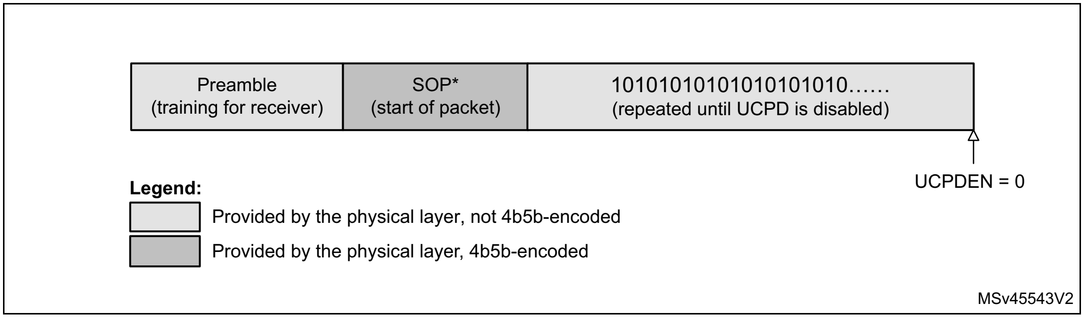
This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 677: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), it is only this programming (and not the block’s behaviour) that differentiates the general packet from the BIST Test Data packet.

BIST Carrier Mode 2

When required, this BIST test mode sends an alternating pattern of 1010 that is continually repeated. As this mode is intended for signal analysis it is stable condition with (in V1.0 of the USB PD specification) no defined length. Starting from V1.1 of the USB PD specification, the protocol layer defines a counter that indicates when to exit this mode.

The way to quit the infinite 1010 sequence (according to requirements of the USB PD specification) is to disable the UCPD peripheral via the UCPDEN bit.

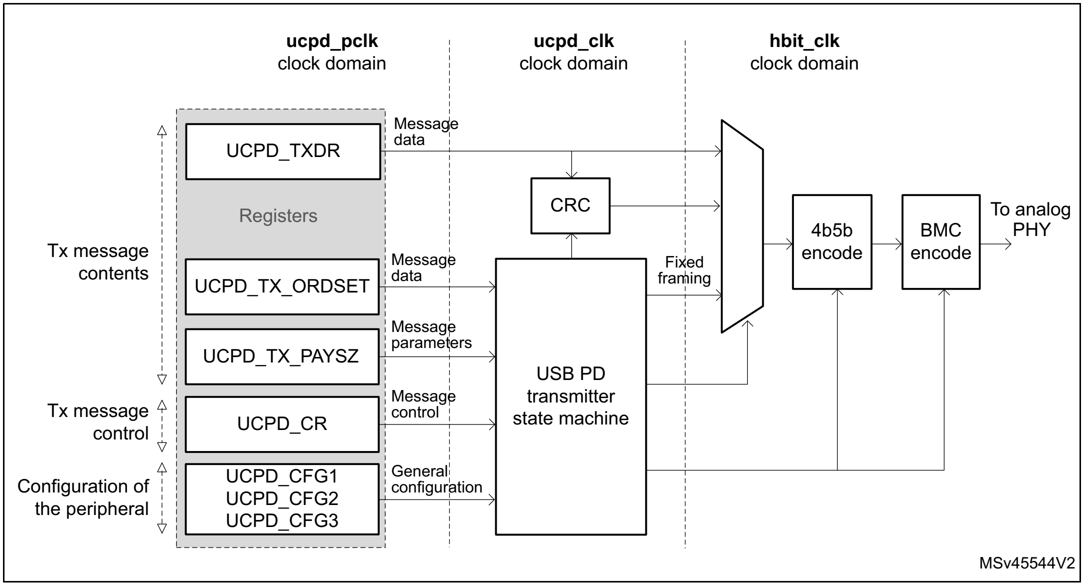
Figure 681. BIST Carrier Mode 2 frame


Preamble
(training for receiver)
SOP*
(start of packet)
10101010101010101010......
(repeated until UCPD is disabled)
UCPDEN = 0
Legend:
Provided by the physical layer, not 4b5b-encoded
Provided by the physical layer, 4b5b-encoded
MSv45543V2

46.4.4 UCPD BMC transmitter

The BMC transmitter comprises 4b5b encoding, CRC generation, and BMC encode, as shown in the following figure. Its output goes to the analog PHY through a channel switch.

Figure 682. UCPD BMC transmitter architecture



BMC encoder

The bi-phase mark coding method is defined in the *IEC 60958-1 Digital Audio Interface Part:1 General Edition 3.0 2008-09 www.iec.ch* specification.

The half-bit clock hbit_clk is derived from ucpd_clk through a simple divider controlled by the HBITCLKDIV[5:0] bitfield of the UCPD_CFGR1 register. This ensures the same duration of high and low half-bit periods (if neglecting a small difference due to different rising and falling edge duration and due to jitter), and the same bit duration (if neglecting jitter).

Transmitter timing and collision avoidance

Hardware support of collision avoidance is made as a function of the half bit time for the transmitter. Two counters are implemented:

- *tInterFrameGap*: via IFRGAP (pre-defined value, can be altered)
- *tTransitionWindow*: via TRANSWIN (pre-defined value, can be altered)

These two counters once set correctly generates the interframe gap.

Hard Reset in transmitter

In order to facilitate generation of a Hard Reset, a special code of TXMODE field is used. No other fields need to be written.

On writing the correct code, the hardware forces Hard Reset Tx under the correct (optimal) timings with respect to an on-going Tx message, which (if still in progress) is cleanly terminated by truncating the current sequence and directly appending an EOP K-code sequence. No specific interrupt is generated relating to this truncation event.

Transmitter behavior in the case of errors

The under-run condition (TXUND interrupt) may happen by accident and in this case, the UCPD is starved of (the correct) Tx payload and is not able to complete the Tx message correctly. This is a serious error (for this to happen the software fails to respond in time). As a result the hardware ensures the CRC is incorrect at the end of the message, thus guaranteeing the message to be discarded at the receiver.

46.4.5 UCPD BMC receiver

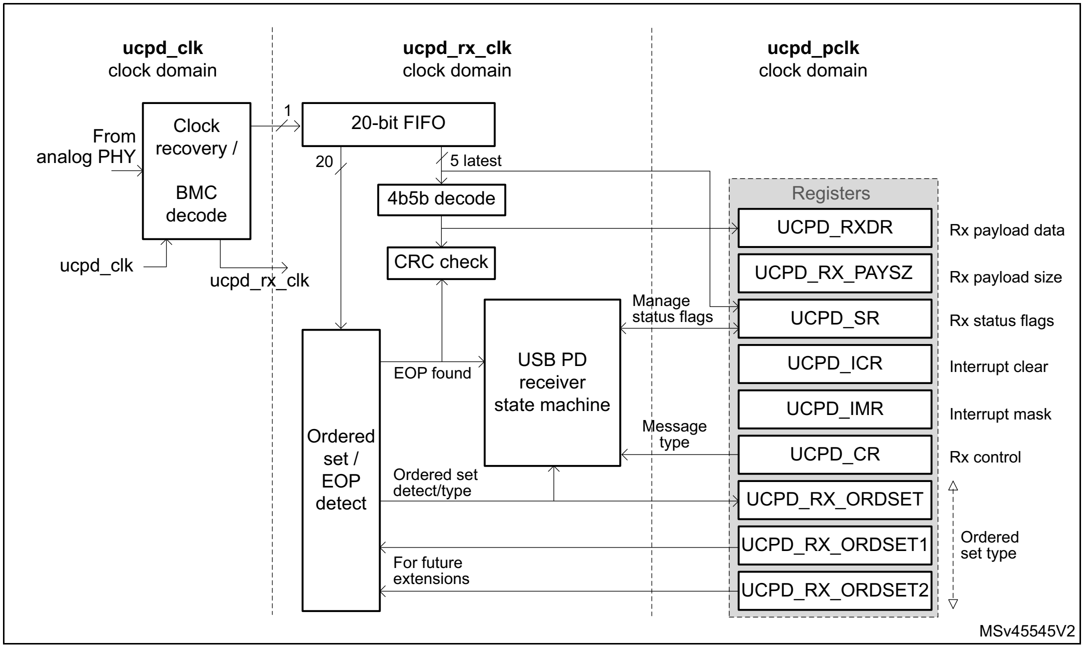
The UCPD BMC receiver performs:

- Clock recovery
- Preamble detection / timing derivation
- BMC decoding
- 4b5b decoding
- K-code ordered set recognition
- CRC checking
- SOP detection
- EOP detection

The receiver is activated as soon as the UCPD peripheral is enabled (via UCPDEN), but it waits for an idle CC line state before attempting to receive a message.

The following figure shows the UCPD BMC receiver high-level architecture.

Figure 683. UCPD BMC receiver architecture



CRC checker

The received bits are fed into a CRC checker which evolves a 32-bit state during the received the payload bitstream. At the end the 32 bits of the CRC also fed into the logic

The EOP detection (5 bits) halts the process and a check is performed for the fixed residual state which confirms correct reception of the payload (in fact the residual is 0xC704DD78).

At this point the UCPD raises interrupt RXMSGEND. If the CRC was not correct then RXERR is set true and the receive data must be discarded.

Under normal operation, this interrupt would previously have been acknowledged and thus cleared. If this is not the case, a different interrupt RXOVR is generated in place of RXMSGEND.

Ordered set detection

This function detects the different ordered sets each consisting of four 5-bit K-codes.

Once we are in the preamble we opens a sliding window detection of the ordered set (4 words of 5 bits).

The ordered sets detected include all SOP* codes (SOP, SOP’, and SOP’’), but also Hard Reset, Cable Reset, SOP’_Debug, SOP’’_Debug, and two extensions defined by registers USBPD1_RX_ORDEXT1 and USBPD1_RX_ORDEXT2.

EOP detection and Hard Reset exception handling

EOP is a fixed 5-bit K-code marking the end of a message.

The way in which a transmitter is required to send a Hard Reset (if a previous message transmit is still in progress) is that this previous message is truncated early with an EOP.

If Hard Reset were ignored, then the EOP detection could be done only at the expected time. However, due to the Hard Reset issue, the EOP detector must be active while an Rx message is arriving. When an “early EOP” is detected, the truncated Rx message is immediately discarded.

Truncated or corrupted message exception

Once the ordered set has been detected, depending on the message, there may be data bytes to be received which is completed with a CRC and EOP. If at any point during these phases an error condition happens:

- the line becomes static for a time significantly longer than one “UI” period (the exact threshold for this condition is not critical but the exception must occur before three UIs), or
- the message goes to the end but it is not recognized (for example EOP is corrupted).

In both cases, the receiver quits the current message, raising RXMSGEND and RXERR flags.

Short preamble or incomplete ordered set exception

In the exceptional case of the receiver seeing less that half of the expected preamble, the frequency estimation allowing correct BMC-decode becomes impossible. Even if the full preamble is seen, allowing frequency estimation, but the ordered set is not fully received before the line becomes static, the receiver state machine does not start.

In both of these cases, the clock-recovery/BMC decoder re-starts, checking initially for an IDLE condition, followed by a preamble, and then estimating frequency.

46.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd)

UCPD offers simple control of these resistors via ANAMODE and ANASUBMODE[1:0]. In case only one of the CC lines is to be used, it is possible to optimize power consumption by disabling control on one the other line, through the CCENABLE[1:0] bitfield.

When the MCU is unpowered, it still presents the “dead battery” Rd, provided that UCPDx_DBCC1 and UCPDx_DBCC2 pins are each connected to UCPDx_CC1 and UCPDx_CC2 pins respectively.

If dead battery behavior is not required (for example for source only products), then UCPDx_DBCC1 and UCPDx_DBCC2 pins must both be tied to ground.

After power arrives and the MCU boots, the desired behavior (for example source) must be programmed into ANAMODE and ANASUBMODE[1:0] before setting the UCPD_DBDIS bit of the PWR_CR3 register to remove dead battery pull-down resistor and allow the values just programmed to take effect.

Use of Standby low-power mode is possible for sinks in the unattached state.

46.4.7 UCPD Type-C voltage monitoring and de-bouncing

For correct operation of the Type-C state machine and for detecting the cable orientation, the CC1/2 lines must be monitored for voltage level, while ignoring fast events such as peaks.

Thresholds between voltage levels on the CC1/2 lines are determined through PHY threshold detector settings.

The TYPEC_VSTATE_CC1/2[1:0] bitfields reflect the CC1/2 line levels processed with a hardware de-bouncing filter that suppresses high-speed line events such as peaks. The PHYCCSEL bit selects the line, CC1 or CC2, to be used for Power Delivery signaling.

For minimizing the power consumption, it is recommended to use the polling method, with the Type-C detectors only turned on for the instant of polling, rather than keeping the Type-C detectors permanently on and wake the device up from Stop mode upon CC1/2 line events.

46.4.8 UCPD fast role swap (FRS) signaling and detection

FRS signaling

The FRS condition (a pulse of a specific length), is generated upon setting the FRSTX bit.

For the duration of FRS condition, the I/O configured as UCPD_FRSTX (alternate function) controls, with high level, the gate of an external NMOS transistor that pulls the active CC line down.

FRS detection

FRS monitoring is enabled by setting the bit FRSRXEN, after writing PHYCCSEL that selects the active CC line depending on the cable orientation detected.

46.4.9 UCPD DMA Interface

DMA is implemented in the UCPD and when it is enabled the byte-level interrupts to handle USBPD1_TXDR and USBPD1_RXDR registers (Tx and Rx data register, each one byte) are no longer needed.

By enabling bits TXDMAEN and/or RXDMAEN, DMA can be activated independently for Tx and/or Rx functionality.

46.4.10 Wakeup from Stop mode

For power consumption optimization, it is useful to use Stop mode and wait for events on CC lines to wake the MCU up.

In order for this to work, it must be first enabled by writing a 1 to WUPEN.

The events causing the wakeup can be:

- Events on the BMC receiver (RXORDDET, RXHRSTDET), hardware enable PHYRXEN
- Event on the FRS detector (FRSEVT), hardware enable FRSRXEN
- Events on the Type-C detectors (TYPECEVT1, TYPECEVT2), hardware enables CC1TCDIS, CC2TCDIS

46.4.11 UCPD programming sequences

The normal sequence of use of the UCPD unit is:

1. Configure UCPD.
2. Enable UCPD.
3. Concurrently:
   - On demand from protocol layer, send Tx message
   - Intercept (poll or wait for interrupt) relevant Rx messages and recover detail to hand off to protocol layer

Repeat the last point infinitely.

Initialization phase

Use the following sequence for a clean startup:

1. Prepare all initial clock divider values, by writing the UCPD_CFG register.
2. Enable the unit, by setting the UCPDEN bit.

Type-C state machine handling

For the general application cases of source, sink, or dual-role port (the last alternating the source and the sink), the software must implement a corresponding USB Type-C state machine. The basic coding is in the following table.

Table 428. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx

ANAMODE	ANASUBMODE[1:0]	Notes	TYPEC_VSTATE_CCx[1:0]			
			00	01	10	11
0: Source	00: Disabled	Disabled	N/A			
	01: Default USB Rp	-	vRa[Def]	vRd[Def]	vOPEN[Def]	N/A
	10: 1.5A Rp	-	vRa[1.5]	vRd[1.5]	vOPEN[1.5]	
	11: 3.0A Rp	-	vRa[3.0]	vRd[3.0]	vOPEN[3.0]	
1: Sink	xx	-	vRa	vRd-USB	vRd-1.5	vRd-3.0

The CCENABLE[1:0] bitfield can disable pull-up/pull-downs on one of the CC lines.

Note: *The Type-C state machine depends not only on CC line levels, but also on VBUS presence detection (sink mode) and when in source mode determines VCONN generation and VBUS state (ON/OFF/+voltage level); discharge). UCPD does not directly control VBUS generation circuitry nor VCONN load switch (enabling VCONN supply generator to be connected to the CC line), but the application needs these inputs and controls to function correctly.*

General programming sequence (with UCPD configured then enabled)

1. Set ANAMODE and ANASUBMODE[1:0] based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This turns on the appropriate pull-ups/pull-downs on the CC lines, and define the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming the PHY is effectively off
2. Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (for example whether the local source is connected to a remote sink)
3. In the case of no connection then wait for a connection event
4. Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages
5. When a new interrupt/event occurs on PHYEVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behaviour from that time] PHYEVT1/2 occurs and the TYPEC_VSTATE1/2 changes accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE and ANASUBMODE[1:0] to start the new behavior

Detailed programming sequence (example):

Table 429. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)

Type-C state	ANAMODE; ANASUBMODE[1:0]	CCENABLE	PHYCCSEL	RDCH	CC[x] VCONNEN[1]	Event => go to next line	Comments
Unattached. SRC	0:Source; 11:Rp3A0	11:both enabled	0 (don't care)	0: [Normal]	00: [neither]	PHYEVT1: [VRd-3A0]	Wait for sink attach detect ; seen on CC1 [EVT1]
Attachwait. SRC			0 [Rd on CC1]			PHYEVT2: [VRa]	Attachwait started (100-200ms) ; now also see the Ra => requesting VCONN
Attached. SRC [VCONN => CC2]	0:Source; 11:Rp3A0 [SinkTxOK]	01: CC2 disable (possible and recommended due to external VCONN switch)			10: [CC2 active]	Timer (100 ms) and no PHYEVTx	Local CC2 disconnected from PHY (VCONN switch connects VCONN source to CC2 externally; Continue to monitor PHYEVT1
	0:Source; 10:Rp1A5 [SinkTxNG]					SW timers (SinkTxNG)	Source wants to initiate message sequence (SinkTxNG condition set first)
	0:Source; 11:Rp3A0 [SinkTxOK]						Source finished message sequence (SinkTxOK condition afterwards)
						PHYEVT1: [VOpen-3A0]	Wait for Sink disconnected (Vopen on CC1)
Unattached wait. SRC		11:both enabled	0 (do not care)	1: [discharge]	00: [neither]	>0.8V detection (or timer?)	Special Source w/VCONN state (ECR Apr 2016): Discharge VCONN [CC2] actively [Rdch] ; to < 0.8V
Unattached. SRC	0:Source; 11:Rp3A0			0: [Normal]			[Details as first line of table]

1. Two GPIOs to enable VCONN through external load switch components

USB PD transmit

On reception of a message from the protocol layer (that is, to be sent), prepare Tx message contents by writing the UCPD_TX_ORDSET and UCPD_TX_PAYSZ registers.

The message transmission is triggered by setting the TXSEND bit, with an appropriate value of the TXMODE bitfield.

When the data byte is transmitted, the TXIS flag is raised to request a new data write to the UCPD_TXDR register.

This re-iterates until the entire payload of data is transmitted.

Upon sending the CRC packet, the TXMSGSENT flag is set to indicate the completion of the message transmission.

Hard Reset transmission

As soon as it is known that a Hard Reset needs to be transmitted, setting the TXHRST bit of the UCPD_CR register forces the internal state machine to generate the correct sequence. The value of UCPD_TX_ORDSET does not require update in this precise case (the correct code for Hard Reset is sent by UCPD).

The USB Power Delivery specification requires that in the case of an ongoing message transmission, the Hard Reset takes precedence. In this case, for example, UCPD truncates the payload of the current message, appending EOP to the end. No notification is available via the registers (for example through the TXMSGSEND flag). This is justified by the fact that the Hard Reset takes precedence over any previous activity (for which it is therefore no longer important to know if it is completed).

Use of DMA for transmission

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the UCPD_CR register.

For each message:

- Prepare the whole message in memory (starting with two header bytes)
- Program the DMA operation with a length corresponding to the two header bytes plus a number of data bytes corresponding to the number of data words multiplied by four
- Write TXSEND to initiate the message transfer
- If TXMSGDISC then go back to previous line (TXSEND)
- Wait for DMA transfer complete interrupt (that is, when last Tx byte written to UCPD)
- Double-check subsequent TXMSGSENT interrupt appears

USB PD receive

Notification of start of the receive message sequence is triggered by an interrupt on UCPD_SR (bit RXORDDET).

The information is recovered by reading:

- UCPD_RX_SOP (on interrupt RXORDDET)
- UCPD_RXDR (on interrupt RXNE, repeats for each byte)
- UCPD_RXPAYSZ (on interrupt RXMSGEND)

The data previously read from UCPD_RXDR above must be discarded at this point if the RXERR flag is set.

If the CRC is valid, the received data is transferred to the protocol layer.

For debug purposes, it may be desirable to track statistics of the number of incorrect K-codes received (this is done only when 3/4 K-codes were valid as defined in the specification). This is facilitated through:

- RXSOP3OF4 bit indicating the presence of at least one invalid K-code
- RXSOPKINVALID bitfield identifying the order of invalid K-code in the ordered set

Use of DMA for reception

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the UCPD_CR register.

Whenever a Rx message is expected:

- Program a DMA receive operation (and spare buffer) a little longer than the maximum possible message (length depends on extended message support).
- After receiving RXORDDET, DMA operation starts working in the background.
- On reception of RXMSGEND interrupt, read RXPAYSZ.
- Double-check RXPAYSZ vs. the number of DMA Rx bytes (must correspond but DMA read of RXDR is slightly after RXDR gets last byte).
- Process the DMA Rx buffer.
- Prepare next Rx DMA buffer as soon as possible in order to be ready.

46.5 UCPD low-power modes

A summary of low-power modes is shown below in *Table 430: Effect of low power modes on the UCPD*.

Table 430. Effect of low power modes on the UCPD

Mode	Description
Sleep	No effect
Stop	Detection of events (Type-C, BMC Rx, FRS detection) remains operational and can wake up the MCU.
Standby	UCPD is not operating, and cannot wake up the MCU. Pull-downs remain active if configured.
Unpowered	Dead battery pull-downs remain active.

The UCPD is able to wakeup the MCU from Stop mode when it recognizes a relevant event, either:

- Type-C event relating to a change in the voltage range seen on either of the CC lines, visible in TYPEC_VSTATE_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wakeup from Stop mode is enabled by setting the WUPEN bit in the UCPD_CFG2 register.

At UCPD level three types of event requiring kernel clock activity may occur during Stop mode:

- Activity on the analog PHY voltage threshold detectors which could later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC line) which could potentially generate an Rx message event (that is, RXORDSET) later
- Activity on Power Delivery FRS detector which could potentially generate an FRS signaling detection event (that is, FRSEVT) later

It order to function correctly with the RCC, the clock request signal is activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC line)
- Power Delivery receiver signal (from the selected CC line)
- FRS detection signal (from the selected CC line)

46.6 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

Table 431. UCPD interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
FRS detection	FRSEVT	Set FRSEVTCF	FRSEVTIE
Type C voltage level change on CC2	TYPECEVT2	Set TYPECEVT2CF	TYPECEVT2IE
Type C voltage level change on CC1	TYPECEVT1	Set TYPECEVT1CF	TYPECEVT1IE
Rx message received	RXMSGEND	Set RXMSGENDCF	RXMSGENDIE
Rx data overflow	RXOVR	Set RXOVRCF	RXOVR
Rx Hard Reset detected	RXHRSTDET	Set RXHRSTDETCF	RXHRSTDETIE
Rx ordered set (4 K-codes) detected	RXORDDET	Set RXORDDETCF	RXORDDETIE
Receive data register not empty	RXNE	Read data in UCPD_RXDR	RXNEIE
Tx data underrun	TXUND	Set TXUNDCF	TXUNDIE
Hard Reset sent	HRSTSENT	Set HRSTSENTCF	HRSTSENTIE
Hard Reset discarded	HRSTDISC	Set HRSTDISCCF	HRSTDISCIE
Transmit message aborted	TXMSGABT	Set TXMSGABTCF	TXMSGABTIE
Transmit message sent	TXMSGSENT	Set TXMSGSENTCF	TXMSGSENTIE
Transmit message discarded	TXMSGDISC	Set TXMSGDISCCF	TXMSGDISCIE
Transmit data required	TXIS	Write data to the UCPD_TXDR register	TXISIE

When an interrupt from the UCPD is received, then the software has to check what is the source of the interrupt by reading the UCPD_SR register.

Depending on which bit is at 1, the ISR must handle that condition and clear the bit by a write to the appropriate bit of the UCPD_ICR register.

46.7 UCPD registers

46.7.1 UCPD configuration register 1 (UCPD_CFGR1)

Address offset: 0x000

Reset value: 0x0000 0000

General configuration of the peripheral. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_USBPDCLK[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**
TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UCPDEN**: UCPD peripheral enable

General enable of the UCPD peripheral.

0: Disable

1: Enable

Upon disabling, the peripheral instantly quits any ongoing activity and all control bits and bitfields default to their reset values. They must be set to their desired values each time the peripheral transits from disabled to enabled state.

Bit 30 **RXDMAEN**: Reception DMA mode enable

When set, the bit enables DMA mode for reception.

0: Disable

1: Enable

Bit 29 **TXDMAEN**: Transmission DMA mode enable

When set, the bit enables DMA mode for transmission.

0: Disable

1: Enable

Bits 28:20 **RXORDSETEN[8:0]**: Receiver ordered set enable

The bitfield determines the types of ordered sets that the receiver must detect. When set/cleared, each bit enables/disables a specific function:

0bxxxxxxxx1: SOP detect enabled

0bxxxxxxx1x: SOP' detect enabled

0bxxxxxx1xx: SOP'' detect enabled

0bxxxxx1xxx: Hard Reset detect enabled

0bxxxx1xxxx: Cable Detect reset enabled

0bxxx1xxxxx: SOP'_Debug enabled

0bxx1xxxxxx: SOP''_Debug enabled

0bx1xxxxxxx: SOP extension#1 enabled

0b1xxxxxxxx: SOP extension#2 enabled

Bits 19:17 **PSC_USBPDCLK[2:0]**: Pre-scaler division ratio for generating ucpd_clk

The bitfield determines the division ratio of a kernel clock pre-scaler producing UCPD peripheral clock (ucpd_clk).

0x0: 1 (bypass)
0x1: 2
0x2: 4
0x3: 8
0x4: 16

It is recommended to use the pre-scaler so as to set the ucpd_clk frequency in the range from 6 to 9 MHz.

Bit 16 Reserved, must be kept at reset value.

Bits 15:11 **TRANSWIN[4:0]**: Transition window duration

The bitfield determines the division ratio (the bitfield value minus one) of a hbit_clk divider producing *tTransitionWindow* interval.

0x00: Not supported
0x01: 2
0x09: 10 (recommended)
0x1F: 32

Set a value that produces an interval of 12 to 20 us, taking into account the ucpd_clk frequency and the HBITCLKDIV[5:0] bitfield setting.

Bits 10:6 **IFRGAP[4:0]**: Division ratio for producing inter-frame gap timer clock

The bitfield determines the division ratio (the bitfield value minus one) of a ucpd_clk divider producing inter-frame gap timer clock (*tInterFrameGap*).

0x00: Not supported
0x01: 2
0x0D: 14
0x0E: 15
0x0F: 16
0x1F: 32

The division ratio 15 is to apply for Tx clock at the USB PD 2.0 specification nominal value. The division ratios below 15 are to apply for Tx clock below nominal, and the division ratios above 15 for Tx clock above nominal.

Bits 5:0 **HBITCLKDIV[5:0]**: Division ratio for producing half-bit clock

The bitfield determines the division ratio (the bitfield value plus one) of a ucpd_clk divider producing half-bit clock (hbit_clk).

0x00: 1 (bypass)
0x1A: 27
0x3F: 64

46.7.2 UCPD configuration register 2 (UCPD_CFGR2)

Address offset: 0x004

Reset value: 0x0000 0000

Configuration of the UCPD Rx signal filtering. Writing to this register is only effective when UCPD is disabled (UCPDEN = 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.	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.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **WUPEN**: Wakeup from Stop mode enable

Setting the bit enables the UCPD_ASYNC_INT signal.

0: Disable

1: Enable

Bit 2 **FORCECLK**: Force ClkReq clock request

0: Do not force clock request

1: Force clock request

Bit 1 **RXFILT2N3**: BMC decoder Rx pre-filter sampling method

Number of consistent consecutive samples before confirming a new value.

0: 3 samples

1: 2 samples

Bit 0 **RXFILTDIS**: BMC decoder Rx pre-filter enable

0: Enable

1: Disable

The sampling clock is that of the receiver (that is, after pre-scaler).

46.7.3 UCPD control register (UCPD_CR)

Address offset: 0x00C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 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.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN
										rw	rw		rw	rs	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
				rw	rw	rw	rw	rw	rw	rw	rw	rs	rs	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC2TCDIS**: CC2 Type-C detector disable

The bit disables the Type-C detector on the CC2 line.
0: Enable
1: Disable
When enabled, the Type-C detector for CC2 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 20 **CC1TCDIS**: CC1 Type-C detector disable

The bit disables the Type-C detector on the CC1 line.
0: Enable
1: Disable
When enabled, the Type-C detector for CC1 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 19 Reserved, must be kept at reset value.

Bit 18 **RDCH**: Rdch condition drive

The bit drives Rdch condition on the CC line selected through the PHYCCSEL bit (thus associated with VCONN), by remaining set during the source-only *UnattachedWait.SRC* state, to respect the Type-C state. Refer to *"USB Type-C ECN for Source VCONN Discharge"*. The CCENABLE[1:0] bitfield must be set accordingly, too.
0: No effect
1: Rdch condition drive

Bit 17 **FRSTX**: FRS Tx signaling enable.

Setting the bit enables FRS Tx signaling.
0: No effect
1: Enable
The bit is cleared by hardware after a delay respecting the USB Power Delivery specification Revision 3.0.

Bit 16 **FRSRXEN**: FRS event detection enable

Setting the bit enables FRS Rx event (FRSEVT) detection on the CC line selected through the PHYCCSEL bit. 0: Disable
1: Enable
Clear the bit when the device is attached to an FRS-incapable source/sink.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:10 **CCENABLE[1:0]**: CC line enable

This bitfield enables CC1 and CC2 line analog PHYs (pull-ups and pull-downs) according to ANAMODE and ANASUBMODE[1:0] setting.
0x0: Disable both PHYs
0x1: Enable CC1 PHY
0x2: Enable CC2 PHY
0x3: Enable CC1 and CC2 PHY
A single line PHY can be enabled when, for example, the other line is driven by VCONN via an external VCONN switch. Enabling both PHYs is the normal usage for sink/source.

Bit 9 **ANAMODE**: Analog PHY operating mode

0: Source
1: Sink
The use of CC1 and CC2 depends on CCENABLE. Refer to *Table 428: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield in conjunction with ANASUBMODE[1:0].

Bits 8:7 **ANASUBMODE[1:0]**: Analog PHY sub-mode

Refer to *Table 428: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield.

Bit 6 **PHYCCSEL**: CC1/CC2 line selector for USB Power Delivery signaling

0: Use CC1 IO for Power Delivery communication
1: Use CC2 IO for Power Delivery communication
The selection depends on the cable orientation as discovered at attach.

Bit 5 **PHYRXEN**: USB Power Delivery receiver enable

0: Disable
1: Enable
Both CC1 and CC2 receivers are disabled when the bit is cleared. Only the CC receiver selected via the PHYCCSEL bit is enabled when the bit is set.

Bit 4 **RXMODE**: Receiver mode

Determines the mode of the receiver.
0: Normal receive mode
1: BIST receive mode (BIST test data mode)
When the bit is set, RXORDSET behaves normally, RXDR no longer receives bytes yet the CRC checking still proceeds as for a normal message.

Bit 3 **TXHRST**: Command to send a Tx Hard Reset

0: No effect
1: Start Tx Hard Reset message
The bit is cleared by hardware as soon as the message transmission begins or is discarded.

Bit 2 **TXSEND**: Command to send a Tx packet

0: No effect
1: Start Tx packet transmission
The bit is cleared by hardware as soon as the packet transmission begins or is discarded.

Bits 1:0 **TXMODE[1:0]**: Type of Tx packet

Writing the bitfield triggers the action as follows, depending on the value:
0x0: Transmission of Tx packet previously defined in other registers
0x1: Cable Reset sequence
0x2: BIST test sequence (BIST Carrier Mode 2)
Others: invalid
From V1.1 of the USB PD specification, there is a counter defined for the duration of the BIST Carrier Mode 2. To quit this mode correctly (after the "tBISTContMode" delay), disable the peripheral (UCPDEN = 0).

46.7.4 UCPD interrupt mask register (UCPD_IMR)

Address offset: 0x010

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 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.	FRSEVTIE	Res.	Res.	Res.	Res.
											r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
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 **FRSEVTIE**: FRSEVT interrupt enable
0: Disable
1: Enable

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2IE**: TYPECEVT2 interrupt enable
0: Disable
1: Enable

Bit 14 **TYPECEVT1IE**: TYPECEVT1 interrupt enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDIE**: RXMSGEND interrupt enable
0: Disable
1: Enable

Bit 11 **RXOVRIE**: RXOVR interrupt enable
0: Disable
1: Enable

Bit 10 **RXHRSTDETIE**: RXHRSTDET interrupt enable
0: Disable
1: Enable

Bit 9 **RXORDDETIE**: RXORDDET interrupt enable
0: Disable
1: Enable

Bit 8 **RXNEIE**: RXNE interrupt enable
0: Disable
1: Enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUNDIE**: TXUND interrupt enable
0: Disable
1: Enable

Bit 5 **HRSTSENTIE**: HRSTSENT interrupt enable
0: Disable
1: Enable

Bit 4 **HRSTDISCIE**: HRSTDISC interrupt enable

0: Disable
1: Enable

Bit 3 **TXMSGABTIE**: TXMSGABT interrupt enable

0: Disable
1: Enable

Bit 2 **TXMSGSENTIE**: TXMSGSENT interrupt enable

0: Disable
1: Enable

Bit 1 **TXMSGDISCIE**: TXMSGDISC interrupt enable

0: Disable
1: Enable

Bit 0 **TXISIE**: TXIS interrupt enable

0: Disable
1: Enable

46.7.5 UCPD status register (UCPD_SR)

Address offset: 0x014

Reset value: 0x0000 0000

The flags (single-bit status bitfields) are associated with interrupt request. Interrupt is generated if enabled by the corresponding bit of the UCPD_IMR register.

31	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.	FRSEVT	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]	
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
r	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVT**: FRS detection event

The flag is cleared by setting the FRSEVTCF bit.
0: No new event
1: New FRS receive event occurred

Bits 19:18 **TYPEC_VSTATE_CC2[1:0]**: CC2 line voltage level

The status bitfield indicates the voltage level on the CC2 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC2 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bits 17:16 **TYPEC_VSTATE_CC1[1:0]**:

The status bitfield indicates the voltage level on the CC1 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC1 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bit 15 **TYPECEVT2**: Type-C voltage level event on CC2 line

The flag indicates a change of the TYPEC_VSTATE_CC2[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.
0: No new event
1: A new Type-C event

Bit 14 **TYPECEVT1**: Type-C voltage level event on CC1 line

The flag indicates a change of the TYPEC_VSTATE_CC1[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.
0: No new event
1: A new Type-C event

Bit 13 **RXERR**: Receive message error

The flag indicates errors of the last Rx message declared (via RXMSGEND), such as incorrect CRC or truncated message (a line becoming static before EOP is met). It is asserted whenever the RXMSGEND flag is set.
0: No error detected
1: Error(s) detected

Bit 12 **RXMSGEND**: Rx message received

The flag indicates whether a message (except Hard Reset message) has been received, regardless the CRC value. The flag is cleared by setting the RXMSGENDCF bit.
0: No new Rx message received
1: A new Rx message received
The RXERR flag set when the RXMSGEND flag goes high indicates errors in the last-received message.

Bit 11 **RXOVR**: Rx data overflow detection

The flag indicates Rx data buffer overflow. It is cleared by setting the RXOVRCF bit.
0: No overflow
1: Overflow
The buffer overflow can occur if the received data are not read fast enough.

Bit 10 **RXHRSTDET**: Rx Hard Reset receipt detection

The flag indicates the receipt of valid Hard Reset message. It is cleared by setting the RXHRSTDETCF bit.

0: Hard Reset not received
1: Hard Reset received

Bit 9 **RXORDDET**: Rx ordered set (4 K-codes) detection

The flag indicates the detection of an ordered set. The relevant information is stored in the RXORDSET[2:0] bitfield of the UCPD_RX_ORDSET register. It is cleared by setting the RXORDDETCF bit.

0: No ordered set detected
1: A new ordered set detected

Bit 8 **RXNE**: Receive data register not empty detection

The flag indicates that the UCPD_RXDR register is not empty. It is automatically cleared upon reading UCPD_RXDR.

0: Rx data register empty
1: Rx data register not empty

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUND**: Tx data underrun detection

The flag indicates that the Tx data register (UCPD_TXDR) was not written in time for a transmit message to execute normally. It is cleared by setting the TXUNDCF bit.

0: No Tx data underrun detected
1: Tx data underrun detected

Bit 5 **HRSTSENT**: Hard Reset message sent

The flag indicates that the Hard Reset message is sent. The flag is cleared by setting the HRSTSENTCF bit.

0: No Hard Reset message sent
1: Hard Reset message sent

Bit 4 **HRSTDISC**: Hard Reset discarded

The flag indicates that the Hard Reset message is discarded. The flag is cleared by setting the HRSTDISCCF bit.

0: No Hard Reset discarded
1: Hard Reset discarded

Bit 3 **TXMSGABT**: Transmit message abort

The flag indicates that a Tx message is aborted due to a subsequent Hard Reset message send request taking priority during transmit. It is cleared by setting the TXMSGABTCF bit.

0: No transmit message abort
1: Transmit message abort

Bit 2 **TXMSGSENT**: Message transmission completed

The flag indicates the completion of packet transmission. It is cleared by setting the TXMSGSENTCF bit.

0: No Tx message completed

1: Tx message completed

In the event of a message transmission interrupted by a Hard Reset, the flag is not raised.

Bit 1 **TXMSGDISC**: Message transmission discarded

The flag indicates that a message transmission was dropped. The flag is cleared by setting the TXMSGDISCCF bit.

0: No Tx message discarded

1: Tx message discarded

Transmission of a message can be dropped if there is a concurrent receive in progress or at excessive noise on the line. After a Tx message is discarded, the flag is only raised when the CC line becomes idle.

Bit 0 **TXIS**: Transmit interrupt status

The flag indicates that the UCPD_TXDR register is empty and new data write is required (as the amount of data sent has not reached the payload size defined in the TXPAYSZ bitfield). The flag is cleared with the data write into the UCPD_TXDR register.

0: New Tx data write not required

1: New Tx data write required

46.7.6 UCPD interrupt clear register (UCPD_ICR)

Address offset: 0x018

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 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.	FRSEVTCF	Res.	Res.	Res.	Res.
											w				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
w	w		w	w	w	w			w	w	w	w	w	w	

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTCF**: FRS event flag (FRSEVT) clear

Setting the bit clears the FRSEVT flag in the UCPD_SR register.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2CF**: Type-C CC2 line event flag (TYPECEVT2) clear

Setting the bit clears the TYPECEVT2 flag in the UCPD_SR register

Bit 14 **TYPECEVT1CF**: Type-C CC1 event flag (TYPECEVT1) clear

Setting the bit clears the TYPECEVT1 flag in the UCPD_SR register

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDCF**: Rx message received flag (RXMSGEND) clear
Setting the bit clears the RXMSGEND flag in the UCPD_SR register.

Bit 11 **RXOVRCF**: Rx overflow flag (RXOVR) clear
Setting the bit clears the RXOVR flag in the UCPD_SR register.

Bit 10 **RXHRSTDETCF**: Rx Hard Reset detect flag (RXHRSTDET) clear
Setting the bit clears the RXHRSTDET flag in the UCPD_SR register.

Bit 9 **RXORDDETCF**: Rx ordered set detect flag (RXORDDET) clear
Setting the bit clears the RXORDDET flag in the UCPD_SR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TXUNDCF**: Tx underflow flag (TXUND) clear
Setting the bit clears the TXUND flag in the UCPD_SR register.

Bit 5 **HRSTSENTCF**: Hard reset send flag (HRSTSENT) clear
Setting the bit clears the HRSTSENT flag in the UCPD_SR register.

Bit 4 **HRSTDISCCF**: Hard reset discard flag (HRSTDISC) clear
Setting the bit clears the HRSTDISC flag in the UCPD_SR register.

Bit 3 **TXMSGABTCF**: Tx message abort flag (TXMSGABT) clear
Setting the bit clears the TXMSGABT flag in the UCPD_SR register.

Bit 2 **TXMSGSENTCF**: Tx message send flag (TXMSGSENT) clear
Setting the bit clears the TXMSGSENT flag in the UCPD_SR register.

Bit 1 **TXMSGDISCCF**: Tx message discard flag (TXMSGDISC) clear
Setting the bit clears the TXMSGDISC flag in the UCPD_SR register.

Bit 0 Reserved, must be kept at reset value.

46.7.7 UCPD Tx ordered set type register (UCPD_TX_ORDSETR)

Address offset: 0x01C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1) and no packet transmission is in progress (TXSEND and TXHRST bits are both low).

31	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.	TXORDSET[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXORDSET[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 **TXORDSET[19:0]**: Ordered set to transmit
The bitfield determines a full 20-bit sequence to transmit, consisting of four K-codes, each of five bits, defining the packet to transmit. The bit 0 (bit 0 of K-code1) is the first, the bit 19 (bit 4 of K-code4) the last.

46.7.8 UCPD Tx payload size register (UCPD_TX_PAYSZR)

Address offset: 0x020

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 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.	TXPAYSZ[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 **TXPAYSZ[9:0]**: Payload size yet to transmit

The bitfield is modified by software and by hardware. It contains the number of bytes of a payload (including header but excluding CRC) yet to transmit: each time a data byte is written into the UCPD_TXDR register, the bitfield value decrements and the TXIS bit is set, except when the bitfield value reaches zero. The enumerated values are standard payload sizes before the start of transmission.

0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)

46.7.9 UCPD Tx data register (UCPD_TXDR)

Address offset: 0x024

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 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.	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]**: Data byte to transmit

46.7.10 UCPD Rx ordered set register (UCPD_RX_ORDSETR)

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.	Res.	RXSOPKINVALID[2:0]			RXSOP3OF4	RXORDSET[2:0]		
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **RXSOPKINVALID[2:0]**:

The bitfield is for debug purposes only.

0x0: No K-code corrupted
0x1: First K-code corrupted
0x2: Second K-code corrupted
0x3: Third K-code corrupted
0x4: Fourth K-code corrupted
Others: Invalid

Bit 3 **RXSOP3OF4**:

The bit indicates the number of correct K-codes. For debug purposes only.

0: 4 correct K-codes out of 4
1: 3 correct K-codes out of 4

Bits 2:0 **RXORDSET[2:0]**: Rx ordered set code detected

0x0: SOP code detected in receiver
0x1: SOP' code detected in receiver
0x2: SOP'' code detected in receiver
0x3: SOP'_Debug detected in receiver
0x4: SOP''_Debug detected in receiver
0x5: Cable Reset detected in receiver
0x6: SOP extension#1 detected in receiver
0x7: SOP extension#2 detected in receiver

46.7.11 UCPD Rx payload size register (UCPD_RX_PAYSZR)

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**
Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **RXPAYSZ[9:0]**: Rx payload size received

This bitfield contains the number of bytes of a payload (including header but excluding CRC) received: each time a new data byte is received in the UCPD_RXDR register, the bitfield value increments and the RXMSGEND flag is set (and an interrupt generated if enabled).

0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)

The bitfield may return a spurious value when a byte reception is ongoing (the RXMSGEND flag is low).

46.7.12 UCPD receive data register (UCPD_RXDR)

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**
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]**: Data byte received

46.7.13 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1)

Address offset: 0x034

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 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.	RXSOPX1[19:16]			
												rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXSOPX1[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 **RXSOPX1[19:0]**: Ordered set 1 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

46.7.14 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2)

Address offset: 0x038

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 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.	RXSOPX2[19:16]			
												rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXSOPX2[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 **RXSOPX2[19:0]**: Ordered set 2 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

46.7.15 UCPD register map

The following table gives the UCPD register map and reset values.

Table 432. UCPD 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	UCPD_CFGR1	UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_USBPDCLK[2:0]			Res.	TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]					
	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
0x004	UCPD_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
	Reset value																													0	0	0	0
0x008	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.
0x00C	UCPD_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN	Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[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

Table 432. UCPD 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
0x010	UCPD_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTIE	Res.	Res.	Res.	Res.	TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
	Reset value												0					0	0		0	0	0	0	0		0	0	0	0	0	0	0
0x014	UCPD_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVT	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]		TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x018	UCPD_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTCF	Res.	Res.	Res.	Res.	TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
	Reset value												0					0	0		0	0	0	0			0	0	0	0	0	0	
0x01C	UCPD_TX _ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXORDSET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	UCPD_TX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	UCPD_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
0x028	UCPD_RX _ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP3OF4	RXORDSET[2:0]		
	Reset value																										0	0	0	0	0	0	0
0x02C	UCPD_RX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x030	UCPD_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
0x034	UCPD_RX _ORDEXTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 432. UCPD 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
0x038	UCPD_RX_ORDEXTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 81* for the register boundary addresses.

47 Debug support (DBG)

47.1 Overview

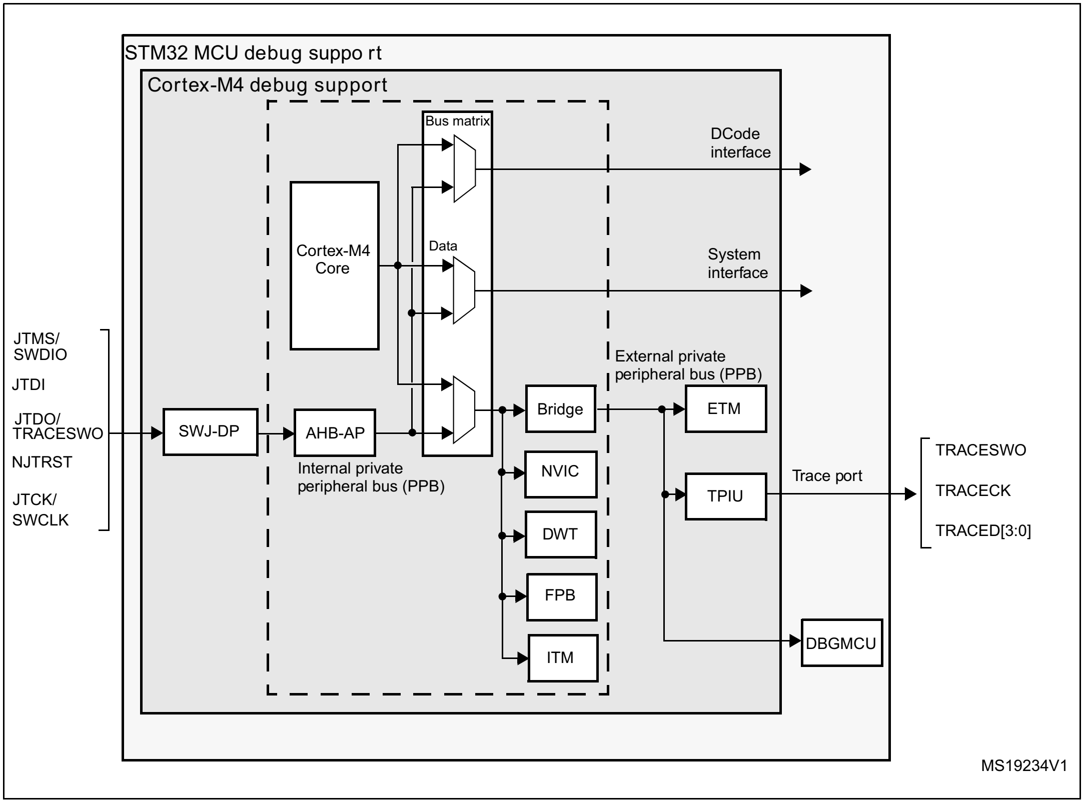
The STM32G4 Series devices are built around a Cortex®-M4 with FPU core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32G4 Series MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

Figure 684. Block diagram of STM32 MCU and Cortex®-M4 with FPU-level debug support



Note: The debug features embedded in the Cortex*®-M4 with FPU core are a subset of the Arm® CoreSight Design Kit.*

The Arm$^{®}$ Cortex$^{®}$-M4 with FPU core provides integrated on-chip debug support. It is comprised of:

- SWJ-DP: Serial wire / JTAG debug port
- AHP-AP: AHB access port
- ITM: Instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface (available on larger packages, where the corresponding pins are mapped)
- ETM: Embedded Trace Macrocell (available only on STM32G4 Series devices larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32G4 Series:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

*Note: For further information on debug functionality supported by the Arm$^{®}$ Cortex$^{®}$-M4 with FPU core, refer to the Cortex$^{®}$-M4 with FPU-*r0p1 *Technical Reference Manual and to the CoreSight Design Kit-r0p1 TRM (see Section 47.2: Reference Arm® documentation).*

47.2 Reference Arm$^{®}$ documentation

- Cortex$^{®}$-M4 with FPU r0p1 Technical Reference Manual (TRM), search for "Cortex$^{®}$-M4 with FPU Technical Reference Manual" at http://infocenter.arm.com
- Arm$^{®}$ Debug Interface V5
- Arm$^{®}$ CoreSight Design Kit revision r0p1 Technical Reference Manual

47.3 SWJ debug port (serial wire and JTAG)

The STM32G4 Series core integrates the Serial Wire / JTAG Debug Port (SWJ-DP). It is an Arm$^{®}$ standard CoreSight debug port that combines a JTAG-DP (5-pin) interface and a SW-DP (2-pin) interface.

- The JTAG Debug Port (JTAG-DP) provides a 5-pin standard JTAG interface to the AHP-AP port.
- The Serial Wire Debug Port (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP port.

In the SWJ-DP, the two JTAG pins of the SW-DP are multiplexed with some of the five JTAG pins of the JTAG-DP.

Figure 685. SWJ debug port

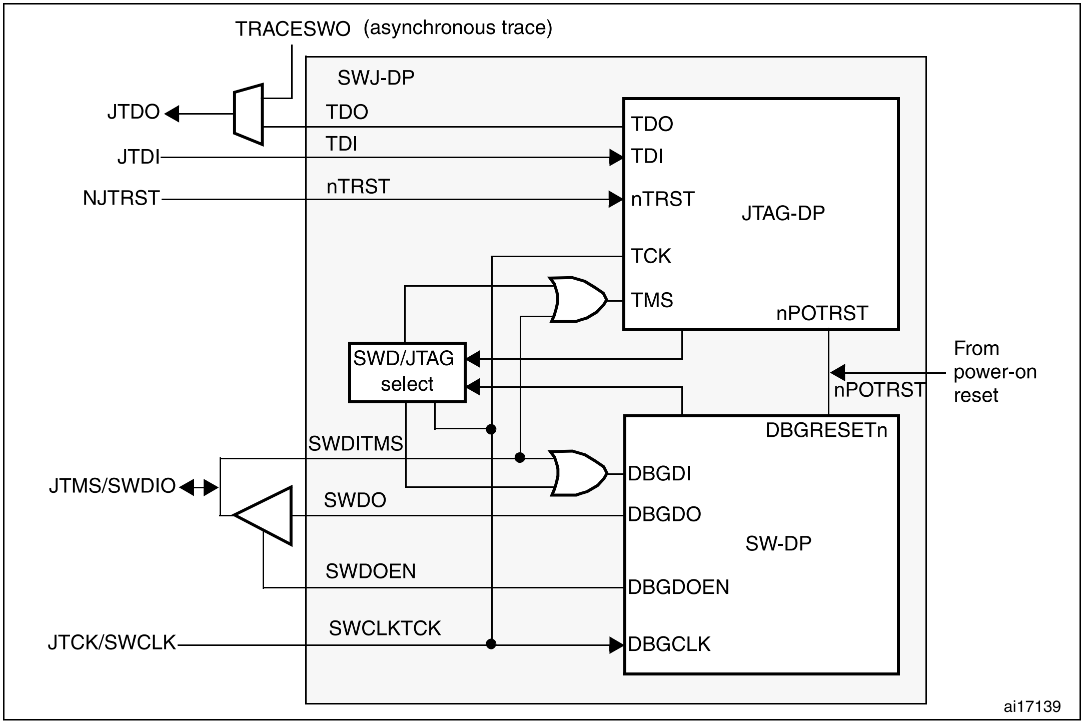


Figure 685 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

47.3.1 Mechanism to select the JTAG-DP or the SW-DP

By default, the JTAG-Debug Port is active.

If the debugger host wants to switch to the SW-DP, it must provide a dedicated JTAG sequence on TMS/TCK (respectively mapped to SWDIO and SWCLK) which disables the JTAG-DP and enables the SW-DP. This way it is possible to activate the SWDP using only the SWCLK and SWDIO pins.

This sequence is:

1. Send more than 50 TCK cycles with TMS (SWDIO) =1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111100111100111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) =1

47.4 Pinout and debug port pins

The STM32G4 Series MCUs are available in various packages with different numbers of available pins. As a result, some functionalities (ETM) related to pin availability may differ between packages.

47.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32G4 Series for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

Table 433. SWJ debug port pins

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assign ment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG Test Mode Selection	IO	Serial Wire Data Input/Output	PA13
JTCK/SWCLK	I	JTAG Test Clock	I	Serial Wire Clock	PA14
JTDI	I	JTAG Test Data Input	-	-	PA15
JTDO/TRACESWO	O	JTAG Test Data Output	-	TRACESWO if asynchronous trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

47.4.2 Flexible SWJ-DP pin assignment

After RESET (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host (note that the trace outputs are not assigned except if explicitly programmed by the debugger host).

However, the STM32G4 Series MCUs offer the possibility of disabling some or all of the SWJ-DP ports, and therefore the possibility of releasing (in gray in the table below) the associated pins for general-purpose I/O (GPIO) usage, except for NJTRST that can be left disconnected but cannot be used as general purpose GPIO without loosing debugger connection. For more details on how to disable SWJ-DP port pins, please refer to *Section 9.3.2: I/O pin alternate function multiplexer and mapping*.

Table 434. Flexible SWJ-DP pin assignment

Available debug ports	SWJ IO pin assigned				
	PA13 / JTMS/ SWDIO	PA14 / JTCK/ SWCLK	PA15 / JTDI	PB3 / JTDO	PB4/ NJTRST
Full SWJ (JTAG-DP + SW-DP) - Reset State	X	X	X	X	X
Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	X	
JTAG-DP disabled and SW-DP enabled	X	X			
JTAG-DP disabled and SW-DP disabled	Released				

47.4.3 Internal pull-up and pull-down on JTAG pins

It is necessary to ensure that the JTAG input pins are not floating since they are directly connected to flip-flops to control the debug mode features. Special care must be taken with the SWCLK/TCK pin which is directly connected to the clock of some of these flip-flops.

To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the JTAG input pins:

- NJTRST: internal pull-up
- JTDI: internal pull-up
- JTMS/SWDIO: internal pull-up
- TCK/SWCLK: internal pull-down

Once a JTAG IO is released by the user software, the GPIO controller takes control again. The reset states of the GPIO control registers put the I/Os in the equivalent state:

- NJTRST: input pull-up
- JTDI: input pull-up
- JTMS/SWDIO: input pull-up
- JTCK/SWCLK: input pull-down
- JTDO: input floating

The software can then use these I/Os as standard GPIOs.

Note: The JTAG IEEE standard recommends to add pull-ups on TDI, TMS and nTRST but there is no special recommendation for TCK. However, for JTCK, the device needs an integrated pull-down.

Having embedded pull-ups and pull-downs removes the need to add external resistors.

The NJTRST (PB4) pin has also UCPD_CC2 functionality which implements internal UCPD pull-down resistor (5.1 KΩ) which is controlled by voltage on UCPD_DBCC2 pin (PA10). In order to use the JTAG, the pull down effect on the CC lines must be removed by using the UCPD1_DBDIS bit (USB Type-C and power delivery dead battery disable) in the PWR_CR3 register (see *Section 46.4.6: UCPD Type-C pull-ups (Rp) and pull-downs (Rd)*).

47.4.4 Using serial wire and releasing the unused debug pins as GPIOs

To use the serial wire DP to release some GPIOs, the user software must change the GPIO (PA15, PB3 and PB4) configuration mode in the GPIO_MODER register.This releases PA15, PB3 and PB4 which now become available as GPIOs.

When debugging, the host performs the following actions:

- Under system reset, all SWJ pins are assigned (JTAG-DP + SW-DP).
- Under system reset, the debugger host sends the JTAG sequence to switch from the JTAG-DP to the SW-DP.
- Still under system reset, the debugger sets a breakpoint on vector reset.
- The system reset is released and the Core halts.
- All the debug communications from this point are done using the SW-DP. The other JTAG pins can then be reassigned as GPIOs by the user software.

Note: For user software designs, note that:

To release the debug pins, remember that they are first configured either in input-pull-up (nTRST, TMS, TDI) or pull-down (TCK) or output tristate (TDO) for a certain duration after reset until the instant when the user software releases the pins.

When debug pins (JTAG or SW or TRACE) are mapped, changing the corresponding IO pin configuration in the IOPORT controller has no effect.

47.5 STM32G4 Series JTAG TAP connection

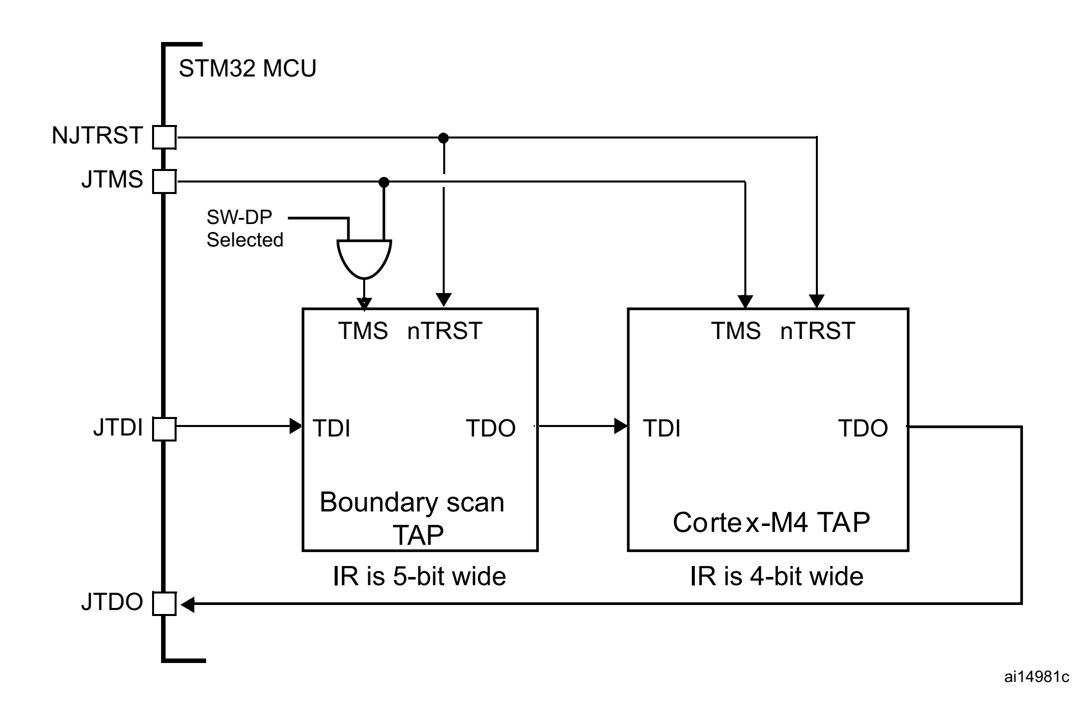
The STM32G4 Series MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex®-M4 with FPU TAP (IR is 4-bit wide).

To access the TAP of the Cortex®-M4 with FPU for debug purposes:

1. First, it is necessary to shift the BYPASS instruction of the boundary scan TAP.
2. Then, for each IR shift, the scan chain contains 9 bits (=5+4) and the unused TAP instruction must be shifted by using the BYPASS instruction.
3. For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

*Note: **Important**: Once Serial-Wire is selected using the dedicated Arm® JTAG sequence, the boundary scan TAP is automatically disabled (JTMS forced high).*

Figure 686. JTAG TAP connections



47.6 ID codes and locking mechanism

There are several ID codes inside the STM32G4 Series MCUs. ST strongly recommends tools designers to lock their debuggers using the MCU DEVICE ID code located in the external PPB memory map at address 0xE0042000.

47.6.1 MCU device ID code

The STM32G4 Series MCUs integrate an MCU ID code. This ID identifies the ST MCU part-number and the die revision. It is part of the DBG_MCU component and is mapped on the external PPB bus (see *Section 47.16 on page 2098*). This code is accessible using the JTAG debug port (4 to 5 pins) or the SW debug port (two pins) or by the user software. It is even accessible while the MCU is under system reset.

Only the DEV_ID(11:0) should be used for identification by the debugger/programmer tools.

DBGMCU_IDCODE

Address: 0xE004 2000

Only 32-bits access supported. Read-only

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 identifier

This field indicates the revision of the device.
0x1000: Revision A
0x1001: Revision Z (Category 4 devices only)
0x2000: Revision B
0x2001: Revision Z
0x2002: Revision Y
0x2003: Revision X

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device identifier

The device ID is:

– 0x468: Category 2 devices (See *Table 1: STM32G4 Series memory density*)
– 0x469: Category 3 devices (See *Table 1: STM32G4 Series memory density*)
– 0x479: Category 4devices (See *Table 1: STM32G4 Series memory density*)

47.6.2 Boundary scan TAP

JTAG ID code

The TAP of the STM32G4 Series BSC (boundary scan) integrates a JTAG ID code equal to 16469041 (category 3 devices), 16468041 (category 2 devices), 16479041 (category 4 devices).

47.6.3 Cortex$^{®}$-M4 with FPU TAP

The TAP of the Arm$^{®}$ Cortex$^{®}$-M4 with FPU integrates a JTAG ID code. This ID code is the Arm$^{®}$ default one and has not been modified. This code is only accessible by the JTAG Debug Port.

This code is **0x4BA00477** (corresponds to Cortex®-M4 with FPU r0p1, see *Section 47.2: Reference Arm® documentation*).

47.6.4 Cortex®-M4 with FPU JEDEC-106 ID code

The Arm® Cortex®-M4 with FPU integrates a JEDEC-106 ID code. It is located in the 4KB ROM table mapped on the internal PPB bus at address 0xE00FF000_0xE00FFFFF.

This code is accessible by the JTAG Debug Port (4 to 5 pins) or by the SW Debug Port (two pins) or by the user software.

47.7 JTAG debug port

A standard JTAG state machine is implemented with a 4-bit instruction register (IR) and five data registers (for full details, refer to the Cortex®-M4 with FPU with FPU r0p1 Technical *Reference Manual (TRM), for references, please see Section 47.2: Reference Arm® documentation)*.

Table 435. JTAG debug port data registers

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	-
1110	IDCODE [32 bits]	ID CODE 0x4BA00477 (Arm® Cortex®-M4 with FPU r0p1-01rel0 ID Code)
1010	DPACC [35 bits]	Debug port access register This initiates a debug port and allows 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 which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved Refer to *Table 436* for a description of the A(3:2) bits

Table 435. JTAG debug port data registers (continued)

IR(3:0)	Data register	Details
1011	APACC [35 bits]	Access port access register Initiates an access port and allows 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 address (sub-address AP registers). Bit 0 = RnW= Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved There are many AP Registers (see AHB-AP) addressed as the combination of: – The shifted value A[3:2] – The current value of the DP SELECT register
1000	ABORT [35 bits]	Abort register – Bits 31:1 = Reserved – Bit 0 = DAPABORT: write 1 to generate a DAP abort.

Table 436. 32-bit debug port registers addressed through the shifted value A[3:2]

Address	A(3:2) value	Description
0x0	00	Reserved, must be kept at reset value.
0x4	01	DP CTRL/STAT register. Used to: – Request a system or debug power-up – Configure the transfer operation for AP accesses – Control the pushed compare and pushed verify operations – Read some status flags (overrun, power-up acknowledges)
0x8	10	DP SELECT register. Used to select the current access port and the active 4-words register window. – Bits 31:24: APSEL: select the current AP – Bits 23:8: reserved – Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP – Bits 3:0: reserved
0xC	11	DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)

47.8 SW debug port

47.8.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 kΩ recommended by Arm®).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

47.8.2 SW protocol sequence

Each sequence consist of three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target

Table 437. Packet request (8-bits)

Bit	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request 1: Read Request
4:3	A(3:2)	Address field of the DP or AP registers (refer to *Table 436*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by the host. Must be read as “1” by the target because of the pull-up

Refer to the Cortex®-M4 with FPU r0p1 *TRM* for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

Table 438. ACK response (3 bits)

Bit	Name	Description
0..2	ACK	001: FAULT 010: WAIT 100: OK

The ACK Response must be followed by a turnaround time only if it is a READ transaction or if a WAIT or FAULT acknowledge has been received.

Table 439. DATA transfer (33 bits)

Bit	Name	Description
0..31	WDATA or RDATA	Write or Read data
32	Parity	Single parity of the 32 data bits

The DATA transfer must be followed by a turnaround time only if it is a READ transaction.

47.8.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default Arm® one and is set to **0x2BA01477** (corresponding to Cortex®-M4 with FPU r0p1).

Note: Note that the SW-DP state machine is inactive until the target reads this ID code.

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the DP has switched from JTAG to SWD or after the line is high for more than 50 cycles
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is **mandatory** to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target issues a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the *Cortex®*-M4 with FPU *r0p1 TRM* and the *CoreSight Design Kit r0p1 TRM*.

47.8.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result.
  The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is “WAIT”. With the exception of

IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.

- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state)
  This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it fails.

47.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 440. SW-DP registers

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read	-	IDCODE	The manufacturer code is not set to ST code **0x2BA01477** (identifies the SW-DP)
00	Write	-	ABORT	-
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: – request a system or debug power-up – configure the transfer operation for AP accesses – control the pushed compare and pushed verify operations. – read some status flags (overrun, power-up acknowledges)
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read	-	READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.
10	Write	-	SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write	-	READ BUFFER	This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction

47.8.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP Registers (see AHB-AP) addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register

47.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP

Features:

- System access is independent of the processor status.
- Either SW-DP or JTAG-DP accesses AHB-AP.
- The AHB-AP is an AHB master into the Bus Matrix. Consequently, it can access all the data buses (Dcode Bus, System Bus, internal and external PPB bus) but the ICode bus.
- Bitband transactions are supported.
- AHB-AP transactions bypass the FPB.

The address of the 32-bits AHP-AP resisters are 6-bits wide (up to 64 words or 256 bytes) and consists of:

c) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register

d) Bits [3:2] = the 2 address bits of A(3:2) of the 35-bit packet request for SW-DP.

The AHB-AP of the Cortex®-M4 with FPU includes 9 x 32-bits registers:

Table 441. Cortex®-M4 with FPU AHB-AP registers

Address offset	Register name	Notes
0x00	AHB-AP Control and Status Word	Configures and controls transfers through the AHB interface (size, hprot, status on current transfer, address increment type
0x04	AHB-AP Transfer Address	-
0x0C	AHB-AP Data Read/Write	-
0x10	AHB-AP Banked Data 0	Directly maps the 4 aligned data words without rewriting the Transfer Address Register.
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	Base Address of the debug interface
0xFC	AHB-AP ID Register	-

Refer to the *Cortex*®-M4 with FPU *r0p1 TRM* for further details.

47.10 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the *Advanced High-performance Bus* (AHB-AP) port. The processor can access these registers directly over the internal *Private Peripheral Bus* (PPB).

It consists of 4 registers:

Table 442. Core debug registers

Register	Description
DHCSR	The 32-bit Debug Halting Control and Status Register: This provides status information about the state of the processor enable core debug halt and step the processor.
DCRSR	The 17-bit Debug Core Register Selector Register: This selects the processor register to transfer data to or from.
DCRDR	The 32-bit Debug Core Register Data Register: This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	The 32-bit Debug Exception and Monitor Control Register: This provides Vector Catching and Debug Monitor Control. This register contains a bit named ***TRCENA*** which enable the use of a TRACE.

Note: ***Important****: these registers are not reset by a system reset. They are only reset by a power-on reset.*

Refer to the *Cortex®-M4 with FPU* r0p*1 TRM* for further details.

To Halt on reset, it is necessary to:

- enable the bit0 (VC_CORRESET) of the Debug and Exception Monitor Control Register
- enable the bit0 (C_DEBUGEN) of the Debug Halting Control and Status Register.

47.11 Capability of the debugger host to connect under system reset

The STM32G4 Series MCUs' reset system comprises the following reset sources:

- POR (power-on reset) which asserts a RESET at each power-up
- Internal watchdog reset
- Software reset
- External reset.

The Cortex®-M4 with FPU differentiates the reset of the debug part (generally PORRESETn) and the other one (SYSRESETn).

This way, it is possible for the debugger to connect under System Reset, programming the Core Debug Registers to halt the core when fetching the reset vector. Then the host can release the system reset and the core immediately halts without having executed any instructions. In addition, it is possible to program any debug features under System Reset.

Note: It is highly recommended for the debugger host to connect (set a breakpoint in the reset vector) under system reset.

47.12 FPB (Flash patch breakpoint)

The FPB unit:

- implements hardware breakpoints
- patches code and data from code space to system space. This feature gives the possibility to correct software bugs located in the Code Memory Space.

The use of a Software Patch or a Hardware Breakpoint is exclusive.

The FPB consists of:

- 2 literal comparators for matching against literal loads from Code Space and remapping to a corresponding area in the System Space
- 6 instruction comparators for matching against instruction fetches from Code Space. They can be used either to remap to a corresponding area in the System Space or to generate a Breakpoint Instruction to the core.

47.13 DWT (data watchpoint trigger)

The DWT unit consists of four comparators. They are configurable as:

- a hardware watchpoint or
- a trigger to an ETM or
- a PC sampler or
- a data address sampler

The DWT also provides some means to give some profiling informations. For this, some counters are accessible to give the number of:

- Clock cycle
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- CPI (clock per instructions)
- Interrupt overhead

47.14 ITM (instrumentation trace macrocell)

47.14.1 General description

The ITM is an application-driven trace source that supports *printf* style debugging to trace *Operating System* (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated as:

- **Software trace.** Software can write directly to the ITM stimulus registers to emit packets.
- **Hardware trace.** The DWT generates these packets, and the ITM emits them.
- **Time stamping.** Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex®-M4 with FPU clock or the bit clock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.

The packets emitted by the ITM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to TPIU) and then output the complete packets sequence to the debugger host.

The bit TRCEN of the Debug Exception and Monitor Control Register must be enabled before you program or use the ITM.

47.14.2 Time stamp packets, synchronization and overflow packets

Time stamp packets encode time stamp information, generic control and synchronization. It uses a 21-bit timestamp counter (with possible prescalers) which is reset at each time stamp packet emission. This counter can be either clocked by the CPU clock or the SWV clock.

A synchronization packet consists of 6 bytes equal to 0x80_00_00_00_00_00 which is emitted to the TPIU as 00 00 00 00 00 80 (LSB emitted first).

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger.

For this, the DWT must be configured to trigger the ITM: the bit CYCCNTENA (bit0) of the DWT Control Register must be set. In addition, the bit2 (SYNCENA) of the ITM Trace Control Register must be set.

Note: If the SYNENA bit is not set, the DWT generates Synchronization triggers to the TPIU which sends only TPIU synchronization packets and not ITM synchronization packets.

An overflow packet consists is a special timestamp packets which indicates that data has been written but the FIFO was full.

Table 443. Main ITM registers

Address	Register	Details
@E0000FB0	ITM lock access	Write 0xC5ACCE55 to unlock Write Access to the other ITM registers
@E0000E80	ITM trace control	Bits 31-24 = Always 0
		Bits 23 = Busy
		Bits 22-16 = 7-bits ATB ID which identifies the source of the trace data
		Bits 15-10 = Always 0
		Bits 9:8 = TSPrescale = Time Stamp Prescaler
		Bits 7-5 = Reserved
		Bit 4 = SWOENA = Enable SWV behavior (to clock the timestamp counter by the SWV clock)
		Bit 3 = DWTENA: Enable the DWT Stimulus
		Bit 2 = SYNCENA: this bit must be to 1 to enable the DWT to generate synchronization triggers so that the TPIU can then emit the synchronization packets
		Bit 1 = TSENA (Timestamp Enable)
		Bit 0 = ITMENA: Global Enable Bit of the ITM
@E0000E40	ITM trace privilege	Bit 3: mask to enable tracing ports31:24
		Bit 2: mask to enable tracing ports23:16
		Bit 1: mask to enable tracing ports15:8
		Bit 0: mask to enable tracing ports7:0
@E0000E00	ITM trace enable	Each bit enables the corresponding Stimulus port to generate trace
@E0000000-E000007C	Stimulus port registers 0-31	Write the 32-bits data on the selected Stimulus Port (32 available) to be traced out

Example of configuration

To output a simple value to the TPIU:

- Configure the TPIU and assign TRACE I/Os by configuring the DBGMCU_CR (refer to *Section 47.17.2: TRACE pin assignment* and *Section 47.16.3: Debug MCU configuration register (DBGMCU_CR)*)
- Write 0xC5ACCE55 to the ITM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00010005 to the ITM Trace Control Register to enable the ITM with Synchronous enabled and an ATB ID different from 0x00
- Write 0x1 to the ITM Trace Enable Register to enable the Stimulus Port 0
- Write 0x1 to the ITM Trace Privilege Register to unmask Stimulus Ports 7:0
- Write the value to output in the Stimulus Port Register 0: this can be done by software (using a printf function)

47.15 ETM (Embedded Trace Macrocell™)

47.15.1 General description

The ETM enables the reconstruction of program execution. Data are traced using the Data Watchpoint and Trace (DWT) component or the Instruction Trace Macrocell (ITM) whereas instructions are traced using the Embedded Trace Macrocell (ETM).

The ETM transmits information as packets and is triggered by embedded resources. These resources must be programmed independently and the trigger source is selected using the Trigger Event Register (0xE0041008). An event could be a simple event (address match from an address comparator) or a logic equation between 2 events. The trigger source is one of the four comparators of the DWT module, The following events can be monitored:

- Clock cycle matching
- Data address matching

For more informations on the trigger resources refer to *Section 47.13: DWT (data watchpoint trigger)*.

The packets transmitted by the ETM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to *Section 47.17: TPIU (trace port interface unit)*) and then outputs the complete packet sequence to the debugger host.

47.15.2 Signal protocol, packet types

This part is described in the section 7 ETMv3 Signal Protocol of the Arm® IHI 0014N document.

47.15.3 Main ETM registers

For more information on registers refer to the chapter 3 of the Arm® IHI 0014N specification.

Table 444. Main ETM registers

Address	Register	Details
0xE0041FB0	ETM Lock Access	Write 0xC5ACCE55 to unlock the write access to the other ETM registers.
0xE0041000	ETM Control	This register controls the general operation of the ETM, for instance how tracing is enabled.
0xE0041010	ETM Status	This register provides information about the current status of the trace and trigger logic.
0xE0041008	ETM Trigger Event	This register defines the event that controls trigger.
0xE004101C	ETM Trace Enable Control	This register defines which comparator is selected.
0xE0041020	ETM Trace Enable Event	This register defines the trace enabling event.
0xE0041024	ETM Trace Start/Stop	This register defines the traces used by the trigger source to start and stop the trace, respectively.

47.15.4 Configuration example

To output a simple value to the TPIU:

- Configure the TPIU and enable the I/IO_TRACEN to assign TRACE I/Os in the STM32G4 Series debug configuration register
- Write 0xC5ACCE55 to the ETM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00001D1E to the control register (configure the trace)
- Write 0000406F to the Trigger Event register (define the trigger event)
- Write 0000006F to the Trace Enable Event register (define an event to start/stop)
- Write 00000001 to the Trace Start/stop register (enable the trace)
- Write 0000191E to the ETM Control Register (end of configuration)

47.16 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog, I^{2}C and bxCAN during a breakpoint
- Control of the trace pins assignment

47.16.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG_SLEEP bit of DBGMCU_CR register must be previously set by the debugger. This feeds HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG_STOP must be previously set by the debugger. This enables the internal RC oscillator clock to feed FCLK and HCLK in Stop mode.
- In Standby mode, the bit DBG_STANDBY must be previously set by the debugger. This keeps the regulators on, and enable the internal RC oscillator clock to feed FCLK and HCLK in Standby mode. A system reset is generated internally so that exiting from Standby is identical than fetching from reset.

The DBGMCU_CR register can be written by the debugger under system reset. If the debugger host does not support these features, it is still possible to write this register by software.

47.16.2 Debug support for timers, RTC, watchdog and I^{2}C

During a breakpoint, it is necessary to choose how the counter of timers,RTC and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the I^{2}C, the user can choose to block the SMBUS timeout during a breakpoint.

The DBGMCU freeze registers can be written by the debugger under system reset. If the debugger host does not support these features, it is still possible to write these registers by software.

47.16.3 Debug MCU configuration register (DBGMCU_CR)

Address: 0xE004 2004

Power-on reset: 0x0000 0000

System reset: not affected

Access: Only 32-bit access 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.	TRACE_MODE [1:0]		TRACE_IOEN	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
								rw	rw	rw			rw	rw	rw

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

Bits 7:5 **TRACE_MODE[1:0] and TRACE_IOEN**: Trace pin assignment control

- With TRACE_IOEN=0:
  TRACE_MODE=xx: TRACE pins not assigned (default state)
- With TRACE_IOEN=1:
  - TRACE_MODE=00: TRACE pin assignment for Asynchronous Mode
  - TRACE_MODE=01: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 1
  - TRACE_MODE=10: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 2
  - TRACE_MODE=11: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 4

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY:** Debug Standby mode

0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.
From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)
1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset.

Bit 1 **DBG_STOP:** Debug Stop mode

0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI16)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc.
1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of DBG_STOP=0)

Bit 0 **DBG_SLEEP:** Debug Sleep mode

0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled.
In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.
1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

47.16.4 Debug MCU APB1 freeze register1 (DBGMCU_APB1FZR1)

Address: 0xE004 2008

Power on reset (POR): 0x0000 0000

System reset: not affected

Access: Only 32-bit access are supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_LPTIM1_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw	rw					rw	rw	rw	rw	rw	rw

Bit 31 **DBG_LPTIM1_STOP:** LPTIM1 counter stopped when core is halted
0: The counter clock of LPTIM1 is fed even if the core is halted
1: The counter clock of LPTIM1 is stopped when the core is halted

Bit 30 **DBG_I2C3_STOP**: I2C3 SMBUS timeout counter stopped when core is halteds halted
0: The Same behavior as in normal mode
1: The I2C3 SMBus timeout is frozen

Bits 29:23 Reserved, must be kept at reset value.

Bit 22 **DBG_I2C2_STOP:** I2C2 SMBUS timeout counter stopped when core is halted
0: Same behavior as in normal mode
1: The I2C2 SMBus timeout is frozen

Bit 21 **DBG_I2C1_STOP:** I2C1 SMBUS timeout counter stopped when core is halted
0: Same behavior as in normal mode
1: The I2C1 SMBus timeout is frozen

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP:** Independent watchdog counter stopped when core is halted
0: The independent watchdog counter clock continues even if the core is halted
1: The independent watchdog counter clock is stopped when the core is halted

Bit 11 **DBG_WWDG_STOP:** Window watchdog counter stopped when core is halted
0: The window watchdog counter clock continues even if the core is halted
1: The window watchdog counter clock is stopped when the core is halted

Bit 10 **DBG_RTC_STOP:** RTC counter stopped when core is halted
0: The clock of the RTC counter is fed even if the core is halted
1: The clock of the RTC counter is stopped when the core is halted

Bits 9:6 Reserved, must be kept at reset value.

Bit 5 **DBG_TIM7_STOP**: TIM7 counter stopped when core is halted
0: The counter clock of TIM7 is fed even if the core is halted
1: The counter clock of TIM7 is stopped when the core is halted

Bit 4 **DBG_TIM6_STOP**: TIM6 counter stopped when core is halted
0: The counter clock of TIM6 is fed even if the core is halted
1: The counter clock of TIM6 is stopped when the core is halted

Bit 3 **DBG_TIM5_STOP**: TIM5 counter stopped when core is halted
0: The counter clock of TIM5 is fed even if the core is halted
1: The counter clock of TIM5 is stopped when the core is halted

Bit 2 **DBG_TIM4_STOP**: TIM4 counter stopped when core is halted
0: The counter clock of TIM4 is fed even if the core is halted
1: The counter clock of TIM4 is stopped when the core is halted

Bit 1 **DBG_TIM3_STOP**: TIM3 counter stopped when core is halted
0: The counter clock of TIM3 is fed even if the core is halted
1: The counter clock of TIM3 is stopped when the core is halted

Bit 0 **DBG_TIM2_STOP:** TIM2 counter stopped when core is halted
0: The counter clock of TIM2 is fed even if the core is halted
1: The counter clock of TIM2 is stopped when the core is halted

47.16.5 Debug MCU APB1 freeze register 2 (DBGMCU_APB1FZR2)

Address: 0xE004 200C

Power on reset (POR): 0x0000 0000

System reset: not affected

Access: Only 32-bit access are 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.	DBG_I2C4_STOP	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DBG_I2C4_STOP:** I2C4 SMBUS timeout counter stopped when core is halted
0: Same behavior as in normal mode
1: The I2C4 SMBus timeout is frozen

Bit 0 Reserved, must be kept at reset value.

47.16.6 Debug MCU APB2 freeze register (DBGMCU_APB2FZR)

Address: 0xE004 2010

Power on reset (POR): 0x0000 0000

System reset: not affected

Access: Only 32-bit access are supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	DBG_HRTIM_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM20_STOP	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP
					rw						rw		rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw		rw											

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **DBG_HRTIM_STOP:** HRTIM counter stopped when core is halted
0: The clock of the HRTIM counter is fed even if the core is halted
1: The clock of the HRTIM counter is stopped when the core is halted

Bits 25:21 Reserved, must be kept at reset value.

Bit 20 **DBG_TIM20_STOP:** TIM20 counter stopped when core is halted

0: The clock of the TIM20 counter is fed even if the core is halted
1: The clock of the TIM20 counter is stopped when the core is halted

Bit19 Reserved, must be kept at reset value.

Bit 18 **DBG_TIM17_STOP:** TIM17 counter stopped when core is halted

0: The clock of the TIM17 counter is fed even if the core is halted
1: The clock of the TIM17 counter is stopped when the core is halted

Bit 17 **DBG_TIM16_STOP:** TIM16 counter stopped when core is halted

0: The clock of the TIM16 counter is fed even if the core is halted
1: The clock of the TIM16 counter is stopped when the core is halted

Bit 16 **DBG_TIM15_STOP:** TIM15 counter stopped when core is halted

0: The clock of the TIM15 counter is fed even if the core is halted
1: The clock of the TIM15 counter is stopped when the core is halted

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **DBG_TIM8_STOP:** TIM8 counter stopped when core is halted

0: The clock of the TIM8 counter is fed even if the core is halted
1: The clock of the TIM8 counter is stopped when the core is halted

Bit 12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP:** TIM1 counter stopped when core is halted

0: The clock of the TIM1 counter is fed even if the core is halted
1: The clock of the TIM1 counter is stopped when the core is halted

Bits 10:0 Reserved, must be kept at reset value.

47.17 TPIU (trace port interface unit)

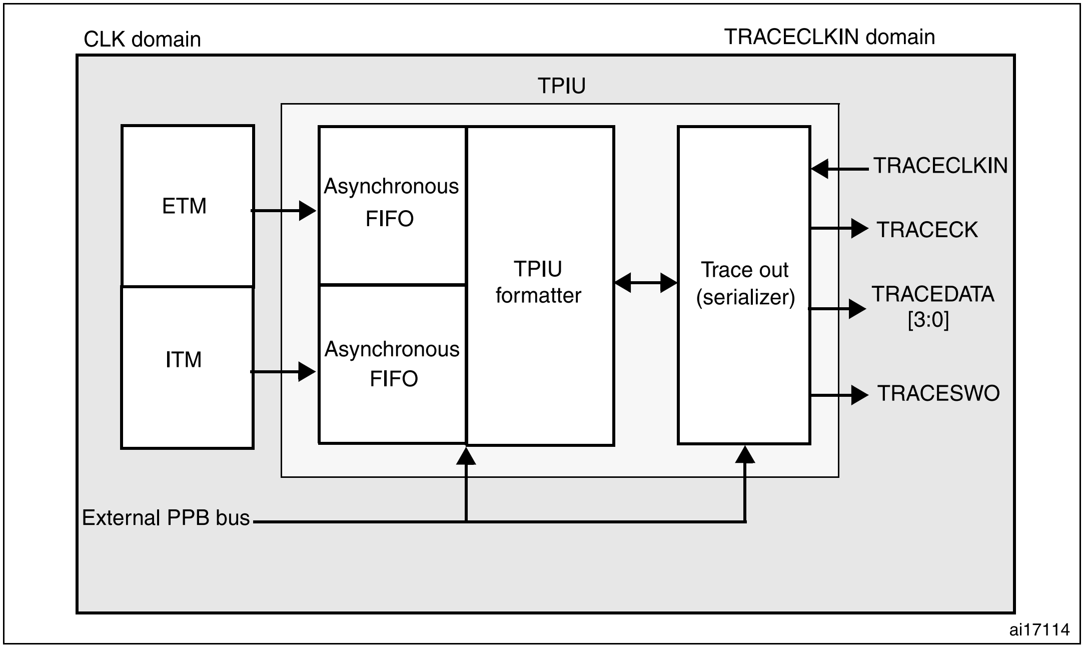
47.17.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM and the ETM.

The output data stream encapsulates the trace source ID, that is then captured by a *trace port analyzer* (TPA).

The core embeds a simple TPIU, especially designed for low-cost debug (consisting of a special version of the CoreSight TPIU).

Figure 687. TPIU block diagram



47.17.2 TRACE pin assignment

- Asynchronous mode
  The asynchronous mode requires 1 extra pin and is available on all packages. It is only available if using Serial Wire mode (not in JTAG mode).

Table 445. Asynchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32G4 Series pin assignment
	Type	Description	
TRACESWO	O	TRACE Asynchronous Data Output	PB3

- Synchronous mode
  The synchronous mode requires from 2 to 6 extra pins depending on the data trace size and is only available in the larger packages. In addition it is available in JTAG mode and in Serial Wire mode and provides better bandwidth output capabilities than asynchronous trace.

Table 446. Synchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32G4 Series pin assignment
	Type	**Description**	
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Synchronous Data Outputs Can be 1, 2 or 4.	PE[6:3]

TPUI TRACE pin assignment

By default, these pins are NOT assigned. They can be assigned by setting the TRACE_IOEN and TRACE_MODE bits in the *Debug MCU configuration register (DBGMCU_CR)*. This configuration has to be done by the debugger host.

In addition, the number of pins to assign depends on the trace configuration (asynchronous or synchronous).

- **Asynchronous mode**: 1 extra pin is needed
- **Synchronous mode**: from 2 to 5 extra pins are needed depending on the size of the data trace port register (1, 2 or 4) :
  - TRACECK
  - TRACED(0) if port size is configured to 1, 2 or 4
  - TRACED(1) if port size is configured to 2 or 4
  - TRACED(2) if port size is configured to 4
  - TRACED(3) if port size is configured to 4

To assign the TRACE pin, the debugger host must program the bits TRACE_IOEN and TRACE_MODE[1:0] of the *Debug MCU configuration register (DBGMCU_CR)*. By default the TRACE pins are not assigned.

This register is mapped on the external PPB and is reset by the PORESET (and not by the SYSTEM reset). It can be written by the debugger under SYSTEM reset.

Table 447. Flexible TRACE pin assignment

DBGMCU_CR register		Pins assigned for:	TRACE IO pin assigned					
TRACE _IOEN	**TRACE _MODE [1:0]**		**PB3** / JTDO/ TRACESWO	**PE2 /** TRACECK	**PE3 /** TRACED[0]	**PE4 /** TRACED[1]	**PE5 /** TRACED[2]	**PE6 /** TRACED[3]
0	XX	No Trace (default state)	Released (1)	-				
1	00	Asynchronous Trace	TRACESWO	-	-	Released (usable as GPIO)		

Table 447. Flexible TRACE pin assignment (continued)

DBGMCU_CR register		**Pins assigned for:**	**TRACE IO pin assigned**					
TRACE _IOEN	**TRACE _MODE [1:0]**		**PB3** / JTDO/ TRACESWO	**PE2 /** TRACECK	**PE3 /** TRACED[0]	**PE4 /** TRACED[1]	**PE5 /** TRACED[2]	**PE6 /** TRACED[3]
1	01	Synchronous Trace 1 bit	Released [1]	TRACECK	TRACED[0]	-	-	-
1	10	Synchronous Trace 2 bit		TRACECK	TRACED[0]	TRACED[1]	-	-
1	11	Synchronous Trace 4 bit		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

[1] When Serial Wire mode is used, it is released, but when JTAG is used, it is assigned to JTDO.

Note: By default, the TRACECLKIN input clock of the TPIU is tied to GND. It is assigned to HCLK two clock cycles after the bit TRACE_IOEN has been set.

The debugger must then program the Trace Mode by writing the PROTOCOL[1:0] bits in the SPP_R (Selected Pin Protocol) register of the TPIU.

- PROTOCOL=00: Trace Port Mode (synchronous)
- PROTOCOL=01 or 10: Serial Wire (Manchester or NRZ) Mode (asynchronous mode). Default state is 01

It then also configures the TRACE port size by writing the bits [3:0] in the CPSPS_R (Current Synchronous Port Size Register) of the TPIU:

- 0x1 for 1 pin (default state)
- 0x2 for 2 pins
- 0x8 for 4 pins

47.17.3 TPUI formatter

The formatter protocol outputs data in 16-byte frames:

- seven bytes of data
- eight bytes of mixed-use bytes consisting of:
  - 1 bit (LSB) to indicate it is a DATA byte (‘0) or an ID byte (‘1).
  - 7 bits (MSB) which can be data or change of source ID trace.
- one byte of auxiliary bits where each bit corresponds to one of the eight mixed-use bytes:
  - if the corresponding byte was a data, this bit gives bit0 of the data.
  - if the corresponding byte was an ID change, this bit indicates when that ID change takes effect.

Note: Refer to the Arm® CoreSight Architecture Specification v1.0 (Arm IHI 0029B) for further information

47.17.4 TPUI frame synchronization packets

The TPUI can generate two types of synchronization packets:

- The Frame Synchronization packet (or Full Word Synchronization packet)

  It consists of the word: 0x7F_FF_FF_FF (LSB emitted first). This sequence can not occur at any other time provided that the ID source code 0x7F has not been used.

  It is output periodically ***between*** frames.

  In continuous mode, the TPA must discard all these frames once a synchronization frame has been found.
- The Half-Word Synchronization packet

  It consists of the half word: 0x7F_FF (LSB emitted first).

  It is output periodically ***between or within*** frames.

  These packets are only generated in continuous mode and enable the TPA to detect that the TRACE port is in IDLE mode (no TRACE to be captured). When detected by the TPA, it must be discarded.

47.17.5 Transmission of the synchronization frame packet

There is no Synchronization Counter register implemented in the TPIU of the core. Consequently, the synchronization trigger can only be generated by the **DWT**. Refer to the registers DWT Control Register (bits SYNCTAP[11:10]) and the DWT Current PC Sampler Cycle Count Register.

The TPUI Frame synchronization packet (0x7F_FF_FF_FF) is emitted:

- after each TPIU reset release. This reset is synchronously released with the rising edge of the TRACECLKIN clock. This means that this packet is transmitted when the TRACE_IOEN bit in the DBGMCU_CFG register is set. In this case, the word 0x7F_FF_FF_FF is not followed by any formatted packet.
- at each DWT trigger (assuming DWT has been previously configured). Two cases occur:
  - If the bit SYNENA of the ITM is reset, only the word 0x7F_FF_FF_FF is emitted without any formatted stream which follows.
  - If the bit SYNENA of the ITM is set, then the ITM synchronization packets follow (0x80_00_00_00_00_00), formatted by the TPUI (trace source ID added).

47.17.6 Synchronous mode

The trace data output size can be configured to 4, 2 or 1 pin: TRACED(3:0)

The output clock is output to the debugger (TRACECK)

Here, TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used.

Note: In this synchronous mode, it is not required to provide a stable clock frequency.

The TRACE I/Os (including TRACECK) are driven by the rising edge of TRACLKIN (equal to HCLK). Consequently, the output frequency of TRACECK is equal to HCLK/2.

47.17.7 Asynchronous mode

This is a low cost alternative to output the trace using only 1 pin: this is the asynchronous output pin TRACESWO. Obviously there is a limited bandwidth.

TRACESWO is multiplexed with JTDO when using the SW-DP pin. This way, this functionality is available in all STM32G4 Series packages.

This asynchronous mode requires a constant frequency for TRACECLKIN. For the standard UART (NRZ) capture mechanism, 5% accuracy is needed. The Manchester encoded version is tolerant up to 10%.

47.17.8 TRACECLKIN connection inside the STM32G4 Series

In the STM32G4 Series, this TRACECLKIN input is internally connected to HCLK. This means that when in asynchronous trace mode, the application is restricted to use time frames where the CPU frequency is stable.

Note: ***Important:*** *when using asynchronous trace: it is important to be aware that:*

The default clock of the STM32G4 Series MCUs is the internal RC oscillator. Its frequency under reset is different from the one after reset release. This is because the RC calibration is the default one under system reset and is updated at each system reset release.

Consequently, the trace port analyzer (TPA) should not enable the trace (with the TRACE_IOEN bit) under system reset, because a synchronization frame packet is issued with a different bit time than trace packets which are transmitted after reset release.

47.17.9 TPIU registers

The TPIU APB registers can be read and written only if the bit TRCENA of the Debug Exception and Monitor Control Register (DEMCR) is set. Otherwise, the registers are read as zero (the output of this bit enables the PCLK of the TPIU).

Table 448. Important TPIU registers

Address	Register	Description
0xE0040004	Current port size	Allows the trace port size to be selected: Bit 0: Port size = 1 Bit 1: Port size = 2 Bit 2: Port size = 3, not supported Bit 3: Port Size = 4 Only 1 bit must be set. By default, the port size is one bit. (0x00000001)
0xE00400F0	Selected pin protocol	Allows the Trace Port Protocol to be selected: Bit1:0 = 00: Synchronous Trace Port Mode 01: Serial Wire Output - manchester (default value) 10: Serial Wire Output - NRZ 11: reserved
0xE0040304	Formatter and flush control	Bit 31-9 = always ‘0’ Bit 8 = TrigIn = always ‘1’ to indicate that triggers are indicated Bit 7-4 = always 0 Bit 3-2 = always 0 Bit 1 = EnFCont. In Synchronous Trace mode (Select_Pin_Protocol register bit1:0 = 00), this bit is forced to ‘1’: the formatter is automatically enabled in continuous mode. In asynchronous mode (Select_Pin_Protocol register bit1:0 <> 00), this bit can be written to activate or not the formatter. Bit 0 = always ‘0’ The resulting default value is 0x102 **Note:** In synchronous mode, because the TRACECTL pin is not mapped outside the chip, the formatter is always enabled in continuous mode; this way the formatter inserts some control packets to identify the source of the trace packets).
0xE0040300	Formatter and flush status	Not used in Cortex®-M4 with FPU, always read as 0x00000008

47.17.10 Example of configuration

- Set the bit TRCENA in the Debug Exception and Monitor Control Register (DEMCR)
- Write the TPIU Current Port Size Register to the desired value (default is 0x1 for a 1-bit port size)
- Write TPIU Formatter and Flush Control Register to 0x102 (default value)
- Write the TPIU Select Pin Protocol to select the synchronous or asynchronous mode. Example: 0x2 for asynchronous NRZ mode (UART like)
- Write the DBGMCU control register to 0x20 (bit IO_TRACEN) to assign TRACE I/Os for asynchronous mode. A TPIU Synchronous packet is emitted at this time (FF_FF_FF_7F)
- Configure the ITM and write the ITM Stimulus register to output a value

47.18 DBG register map

The following table summarizes the Debug registers

Table 449. DBG register map and reset values

Addr.	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
0xE0042000	DBGMCU_ IDCODE	REV_ID																Res.	Res.	Res.	Res.	DEV_ID											
	Reset value[1]	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
0xE0042004	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.	TRACE_MODE[1:0]		TRACE_IOEN	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value																									0	0	0			0	0	0
0xE004 2008	DBGMCU_ APB1FZR1	DBG_LPTIM1_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value	0	0								0	0									0	0	0					0	0	0	0	0	0
0xE004 200C	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.	Res.	Res.	Res.	Res.	Res.	DBG_I2C4_STOP	Res.
	Reset value																																
0xE004 2010	DBGMCU_ APB2FZR	Res.	Res.	Res.	Res.	Res.	DBG_HRTIM_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM20_STOP	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value						0						0		0	0	0			0		0											

1. The reset value is product dependent. For more information, refer to *Section 47.6.1: MCU device ID code*.

48 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 to the characteristics of the STM32G4 Series microcontroller.

48.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF 7590

Address offset: 0x00

Read only = 0xXXXX XXXX where 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

Address offset: 0x04

Read only = 0xXXXX XXXX where 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

Read only = 0xXXXX XXXX where 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)

48.2 Flash size data register

Base address: 0x1FFF 75E0

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE															
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
This bitfield indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x040 corresponds to 64 Kbytes.

48.3 Package data register

Base address: 0x1FFF 7500

Address offset: 0x00

Read only = 0xXXXX where X is factory-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: LQFP64
00001: WLCSP64
00010: LQFP100 (all devices) and LQFP80 (for category 2 and category 3 devices)
00101: WLCSP81
00111: LQFP128 / UFBGA121
01000: UFQFPN32
01001: LQFP32
01010: UFQFPN48
01011: LQFP48
01100: WLCSP49
01101: UFBGA64
01110: TFBGA100
10001: LQFP80 (for category 4 devices only)
Others: reserved

49 Revision history

Table 450. Document revision history

Date	Revision	Changes
06-May-2019	1	Initial release.
10-Oct-2019	2	**Document convention section** – Updated *Table 2: Product specific features*. **System architecture section** – Updated *Section 2.1: System architecture* replacing FMC by FSMC. – Updated *Figure 2: Memory map* replacing FMC by FSMC. **Memory organization section** – Updated *Table 3: STM32G4 Series memory map and peripheral register boundary addresses* – Updated *Table 2.4: Embedded SRAM* SRAM2 (mapped at address 0x2000 4000). – Updated *Table 7: Flash module - 512/256/128 KB dual bank organization (64 bits read width)*. – Updated *Table 8: Flash module - 512/256/128 KB single bank organization (128 bits read width)*. **Power control section** – Updated *Section 6.1: Power supplies*. **Reset and clock control section** – Updated *Figure 17: Clock tree*. Updated FMC into FSMC in: – *Section 7.4.10: AHB3 peripheral reset register (RCC_AHB3RSTR)*. – *Section 7.4.16: AHB3 peripheral clock enable register(RCC_AHB3ENR)*. – *Section 7.4.22: AHB3 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR)*. **System controller configuration section** – Updated *Section 10.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* bits[2:0] description. **Peripherals interconnect matrix section** – Updated *Table 82: Interconnect 11* on-chip source FLTxSRC[1:0] = 01 column. **Nested vectored interrupt controller section** – Updated *Table 97: STM32G4 Series vector table*. **Flexible static memory controller (FSMC) section** – Updated *Section 19: Flexible static memory controller (FSMC)*. **CORDIC co-processor (CORDIC) section** – Updated *Section 17: CORDIC co-processor (CORDIC)*. **Analog digital converter (ADC) section** – Updated *Section 21.4.33: Monitoring the internal voltage reference*. – Updated *Section : Sampling time control trigger mode*.

Table 450. Document revision history (continued)

Date	Revision	Changes
10-Oct-2019	2 (continued)	**Digital analog converter (DAC) section** – Updated *Section 22: Digital-to-analog converter (DAC)*. **Comparator section** – Updated *Section 24.6.2: COMP register map*. **AES hardware accelerator (AES)** - Updated *Section 34: AES hardware accelerator (AES)*. **USB power delivery interface (UCPD)section** – Updated *Section 46: USB Type-C™ / USB Power Delivery interface (UCPD)*. **Debug section** – Updated *Section 47.4.2: Flexible SWJ-DP pin assignment*. – Updated *Section 47.8.3: SW-DP state machine (reset, idle states, ID code)* ID code by 0x2BA01477. – Updated *Table 447: Flexible TRACE pin assignment*. – Updated *Table 435: JTAG debug port data registers*.
26-Mar-2020	3	Added Category 4 devices (STM32G491, STM32G4A1) in: – *Table 1: STM32G4 Series memory density*. – *Table 2: Product specific features*. – *Section 2.4: Embedded SRAM*. – *Section 2.4.1: Parity check*. – *Table 4: CCM SRAM organization*. – *Section 2.5: Flash memory overview*. – *Section 4: Embedded Flash memory (FLASH) for category 4 devices*. – *Table 90: DMAMUX instantiation*. **Embedded Flash memory (FLASH) section:** Updated: – *Number of wait states according to CPU clock (HCLK) frequency* tables for all categories. – *Table 11: Option byte organization*. – *Table 21: Option byte organization*. – *Section 3.4.2: Option bytes programming* ‘activating dual bank mode (switching from DBANK=0 to DBANK=1)’ paragraph. – *Section 4.7.6: Flash control register (FLASH_CR)* PNB[7:0] bits. – ‘User and read protection option bytes’ and *Section 4.7.8: Flash option register (FLASH_OPTR)* register description adding PB4_PUPEN bit. – *Section 5.7.14: FLASH register map*. **Power control section:** Updated: – *Table 38: Range 1 boost mode configuration* removing the lower SYSCLK limits. – *Section 6.4.2: Power control register 2 (PWR_CR2)* PLS[2:0] bit description. **Reset and clock control section:** Updated: – *Figure 17: Clock tree*. – *Table 51: RCC register map and reset values*. **Peripherals interconnect matrix section:** – Updated *Table 60: STM32G4 Series peripherals interconnect matrix*.

Table 450. Document revision history (continued)

Date	Revision	Changes
26-Mar-2020	3 (continued)	**DMA request multiplexer (DMAMUX) section:** – Updated *Section 13.3.2: DMAMUX mapping*; **Analog digital converter section:** Updated: – *Section 21.2: ADC main features*. – *Figure 83: ADC clock scheme*. – *Figure 86: ADC3 connectivity*. – *Section 21.4.7: Single-ended and differential input channels*. **Comparator section:** Updated *Figure 168: Comparator block diagram*. **Operational amplifier section:** Updated: – *Table 200: Operational amplifier possible connection*. – *Section 25.3.7: Calibration* procedure. – *Section 25.3.8: Timer controlled Multiplexer mode* procedure. – *Section 25.5.1: OPAMP1 control/status register (OPAMP1_CSR)*. – *Section 25.5.2: OPAMP2 control/status register (OPAMP2_CSR)*. – *Section 25.5.3: OPAMP3 control/status register (OPAMP3_CSR)*. – *Section 25.5.4: OPAMP4 control/status register (OPAMP4_CSR)*. – *Section 25.5.5: OPAMP5 control/status register (OPAMP5_CSR)*. – *Section 25.5.6: OPAMP6 control/status register (OPAMP6_CSR)*. **FD controller area network section:** Updated *Section 44.4.7: FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)* note. **Debug support section:** Updated: – *Section 47.4.3: Internal pull-up and pull-down on JTAG pins*. – *Section 47.6.1: MCU device ID code*. – *Section 47.6.2: Boundary scan TAP*. **Device electronic signature section:** – Added *Section 48.3: Package data register*.

Table 450. Document revision history (continued)

Date	Revision	Changes
14-Apr-2020	4	**Embedded Flash memory (FLASH) section:** Updated: – *Table 9: Number of wait states according to CPU clock (HCLK) frequency*. – *Table 19: Number of wait states according to CPU clock (HCLK) frequency*. – *Table 29: Number of wait states according to CPU clock (HCLK) frequency*. – *Section 5.4.1: Option bytes description* 'Securable memory area option bytes' paragraph. – *Section 5.7.13: Flash Securable area register (FLASH_SEC1R)*. **Reset and clock control (RCC) section:** Updated: – *Section 7.2.4: PLL*. – *Section 7.4.4: PLL configuration register (RCC_PLLCFGR)* PLLN[6:0] and PLLM[3:0] description. **High-resolution timer (HRTIM) section:** – Updated *Section 27.3.1: General description*.
20-Nov-2020	5	**Memory map section:** Updated *Table 3: STM32G4 Series memory map and peripheral register boundary addresses* USB SRAM to 1 Kbyte. Updated *Figure 2: Memory map*. **Embedded Flash section:** Updated for category 3 devices: – 'User and read protection option bytes' register bit 29,28 name at NRST_MODE. – 'Securable memory area Bank 1 option bytes' register BOOT_LOCK description removing caution. – *Section 3.7.17: Flash Securable area bank1 register (FLASH_SEC1R)* BOOT_LOCK description. Updated for category 4 devices: – 'User and read protection option bytes' register bit 29,28 name at NRST_MODE. – 'Securable memory area option bytes' register BOOT_LOCK description removing caution. – *Section 4.7.1: Flash access control register (FLASH_ACR)* reset value. – *Section 4.7.13: Flash Securable area register (FLASH_SEC1R)* BOOT_LOCK description. – *Section 4.7.14: FLASH register map* Updated for category 2 devices: – 'User and read protection option bytes' register bit 29,28 name at NRST_MODE. – 'Securable memory area option bytes' register BOOT_LOCK description removing caution. – *Section 5.7.13: Flash Securable area register (FLASH_SEC1R)* BOOT_LOCK description. Updated 'rw' to 'r' for all option bytes.

Table 450. Document revision history (continued)

Date	Revision	Changes
20-Nov-2020	5 (continued)	Replaced ‘default’ by ‘production value’ for RDP level 0 instead of RDP level 1 in: – *Table 12: Flash memory read protection status*. – *Figure 4: Changing the read protection (RDP) level*. – *Table 22: Flash memory read protection status*. – *Figure 7: Changing the read protection (RDP) level*. – *Table 32: Flash memory read protection status*. – *Figure 10: Changing the read protection (RDP) level*. **Power control (PWR) section**: Updated *Section 6.4.22: Power control register (PWR_CR5)*. **Reset and clock control (RCC) section:** Updated – *Section 7.2.11: ADC clock*. – *Section 7.4.19: APB2 peripheral clock enable register (RCC_APB2ENR)* SPI4EN bit description. **Peripherals interconnect matrix section**: – Updated *Table 73: Interconnect 2*. **DMA request multiplexer (DMAMUX) section:** Updated: – *Section 13.3.2: DMAMUX mapping*. – *Section 13.6.1: DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)*. **Nested vectored interrupt controller (NVIC) section:** Updated *Table 97: STM32G4 Series vector table*. **Analog-to-digital converters (ADC) section:** Updated *Section 21.2: ADC main features*. **High-resolution timer (HRTIM) section:** Updated: – *Table 221: EExFLTR[3:0] codes depending on UDM bit setting*. – *Table 222: External events features*. – *Section 27.3.8: External events global conditioning*. – *Section 27.3.9: External event filtering in timing units*. – *Section 27.5.28: HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F)*. – *Section 27.5.66: HRTIM ADC trigger 1 register (HRTIM_ADC1R)*. – *Section 27.5.67: HRTIM ADC trigger 2 register (HRTIM_ADC2R)*. – *Section 27.5.68: HRTIM ADC trigger 3 register (HRTIM_ADC3R)* adding note. **General-purpose timers (TIM2/TIM3/TIM4/TIM5) section:** Updated *Table 285: TIM2/TIM3/TIM4/TIM5 register map and reset values*. **General purpose timers (TIM15/TIM16/TIM17):** Updated: – *Section 30.8.19: TIMx option register 1 (TIMx_OR1)(x = 16 to 17)* address offset to 0x68. – *Section 30.8.22: TIM16/TIM17 register map*. **Debug support (DBG) section:** – Updated *Section 47.6.1: MCU device ID code* REV_ID[15:0] bits description.

Table 450. Document revision history (continued)

Date	Revision	Changes
09-Feb-2021	6	**Embedded Flash section:** Updated: – *Section 3.5.3: Write protection (WRP)*. – *Section 3.5.4: Securable memory area*. – *Section 3.7.1: Flash access control register (FLASH_ACR)* reset value – *Section 4.5.3: Write protection (WRP)*. – *Section 4.5.4: Securable memory area*. – *Section 5.4.1: Option bytes description* securable memory area option bytes paragraph. – *Section 5.5.3: Write protection (WRP)*. – *Section 5.5.4: Securable memory area*. – *Section 5.7.1: Flash access control register (FLASH_ACR)* reset value. **Filter math accelerator (FMAC) section:** Updated *Section 18.4: FMAC registers* **Analog digital converter section:** Updated: – *Table 177: ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC)* ADC_CFGR2 address. – *Section 21.6.6: ADC sample time register 1 (ADC_SMPR1)* SMPPLUS bit. **Operational amplifier (OPAMP) section:** – Updated *Table 200: Operational amplifier possible connection*. **USB Type-C™ / USB Power Delivery interface (UCPD) section:** Updated: – *Section 46.7.3: UCPD control register (UCPD_CR)*. – *Section 46.7.15: UCPD register map*. Removed UCPD configuration register 3 (UCPD_CFGR3). **Debug support (DBG) section:** Updated *Section 47.4.2: Flexible SWJ-DP pin assignment* removing note.

Index

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2021 STMicroelectronics – All rights reserved