RM0438
Reference manual

STM32L552xx and STM32L562xx advanced Arm®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32L552xx and STM32L562xx microcontrollers memory and peripherals.

STM32L552xx and STM32L562xx belong to the STM32L5x2 line of microcontrollers with different memory sizes, packages and peripherals.

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

For information on the Arm® Cortex®-M33 core, refer to the Cortex®-M33 *Technical Reference manual*.

Related documents

- Cortex®-M33 *Technical Reference Manual* available at http://infocenter.arm.com
- STM32L552xx and STM32L562xx datasheets

Contents

List of tables

List of figures

1 Documentation conventions

1.1 General information

The STM32L552xx and STM32L562xx devices have an Arm®(a) Cortex®-M33 core.

arm

1.2 List of abbreviations for registers

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

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

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

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

1.3 Glossary

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

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **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.
- **OBL**: option byte loader.
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.
- **RAZ**: read-as-zero.
- **WI**: writes ignored.
- **RAZ/WI**: read-as-zero, writes ignored.

1.4 Availability of peripherals

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

2 Memory and bus architecture

The STM32L552xx and STM32L562xx devices are ultra-low-power microcontrollers (STM32L5 Series) based on the high-performance Arm® Cortex®-M33 32-bit RISC core. They operate at frequencies up to 110 MHz.

The Cortex®-M33 processor delivers a high computational performance with low-power consumption and an advanced response to interrupts. it features:

- Arm® TrustZone® technology, using the Armv8-M main extension supporting secure and non-secure states
- Memory protection units (MPUs), supporting 8 regions for secure world and 8 regions for non-secure world.
- Configurable security attribution unit (SAU) supporting up to 8 memory regions
- Floating-point arithmetic functionality with support for single precision arithmetic

The Cortex®-M33 processor supports the following bus interfaces:

- System AHB bus: the system AHB (S-AHB) bus interface is used for any instruction fetch and data access to the memory-mapped SRAM, peripherals, external RAM and external devices, or Vendor_SYS regions of the Armv8-M memory map.
- Code AHB bus: the Code AHB (C-AHB) bus interface is used for any instruction fetch and data access to the code region of the Armv8-M memory map.

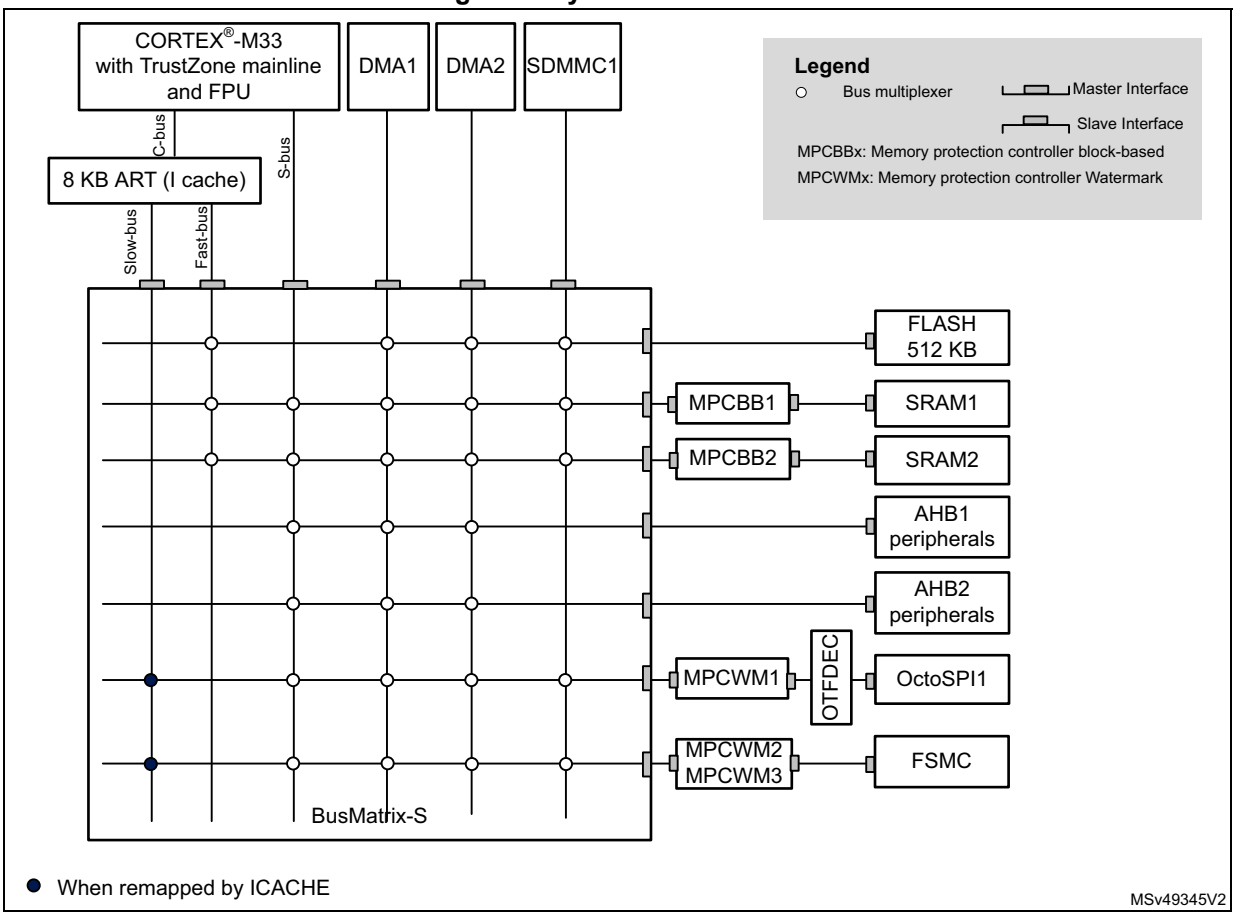
2.1 System architecture

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

- Up to six masters:
  - Fast C-bus, connecting Cortex®-M33 with TrustZone and FPU core C-bus to the internal memories through the ART (instruction cache)
  - Slow C-bus, connecting Cortex®-M33 with TrustZone and FPU core C-bus to the external memories through the ART (instruction cache)
  - Cortex®-M33 with TrustZone and FPU core S-bus
  - DMA1
  - DMA2
  - SDMMC1 bus
- Up to seven slaves:
  - Internal Flash memory on the fast C-bus
  - Internal SRAM1 (192 Kbytes)
  - Internal SRAM2 (64 Kbytes)
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - Flexible static memory controller (FSMC)
  - OCTOSPI1 memory controller

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 Fast C-bus

This bus connects the C-bus of the Cortex®-M33 core to the BusMatrix via the instruction cache. This bus is used for instruction fetch and data access to the internal memories mapped in code region. The target of this bus are the internal Flash and internal SRAMs.

2.1.2 Slow C-bus

This bus connects the C-bus of the Cortex®-M33 core to the BusMatrix via the instruction cache. This bus is used for instruction fetch and data access to the external memories mapped in code region. The target of this bus are the external memories (FSMC and OCTOSPI).

2.1.3 S-bus

This bus connects the system bus of the Cortex®-M33 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 SRAMs, the AHB1 peripherals including the APB1 and APB2

peripherals, the AHB2 peripherals and the external memories through the OCTOSPI or the FSMC.

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

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 and SRAM2, the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals and the external memories through the OCTOSPI or the FSMC.

2.1.5 SDMMC controller DMA bus

This bus connects the SDMMC1 DMA master interface to the BusMatrix. This bus is used only by the SDMMC1 DMA to load/store data from/to memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2), internal Flash memory or external memories through FSMC or OCTOSPI.

2.1.6 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a Round Robin algorithm. The BusMatrix is composed of up to six masters (CPU AHB, system bus, Fast C-bus, Slow C-bus, DMA1, DMA2, SDMMC1) and up to seven slaves (FLASH, SRAM1, SRAM2, AHB1 (including APB1 and APB2), AHB2, OCTOSTPI1 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 2.3.2: Memory map and register boundary addresses on page 88* 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 the user has 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 TrustZone® security architecture

The security architecture is based on Arm® TrustZone® with the Armv8-M Main Extension.

The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register.

When the TrustZone is enabled, the SAU (security attribution unit) and IDAU (implementation defined attribution unit) define the memory access permissions based on secure or non-secure state of the processor.

- SAU: Up to eight SAU configurable regions are available for security attribution.
- IDAU: It provides a first memory partition between non-secure and non-secure-callable regions. The IDAU memory map partition is not configurable and fixed by hardware

implementation (refer to *Figure 2*). It is then combined with the results from the SAU security attribution and the most secure level between the two is selected.

Based on IDAU security attribution, the Flash, system SRAMs and peripherals memory space is aliased twice for secure and non-secure state. However, the external memories space is not aliased.

Table 1 shows an example of typical eight SAU regions mapping based on IDAU regions.

The user can split and choose the secure, non-secure or NSC regions for external memories as needed.

Table 1. Example of memory map security attribution versus SAU regions configuration(1)

Region description	Address range	IDAU security attribution	SAU security attribution typical configuration	Final security attribution
Code - external memories	0x0000_0000 0x07FF_FFFF	Non-secure	Secure or non-secure or NSC	Secure or non-secure or NSC
Code - Flash and SRAM	0x0800_0000 0x0BFF_FFFF	Non-secure	Non-secure	Non-secure
	0x0C00_0000 0x0FFF_FFFF	NSC	Secure or NSC	Secure or NSC
Code - external memories	0x1000_0000 0x17FF_FFFF	Non-secure	Non-secure	Non-secure
	0x1800_0000 0x1FFF_FFFF	Non-secure	Non-secure	Non-secure
SRAM	0x2000_0000 0x2FFF_FFFFF	Non-secure	Non-secure	Non-secure
	0x3000_0000 0x3FFF_FFFFF	NSC	Secure or NSC	Secure or NSC
Peripherals	0x4000_0000 0x4FFF_FFFFF	Non-secure	Non-secure	Non-secure
	0x5000_0000 0x5FFF_FFFFF	NSC	Secure or NSC	Secure or NSC
External memories	0x6000_0000 0xDFFF_FFFF	Non-secure	Secure or non-secure or NSC	Secure or non-secure or NSC

1. Different colors highlights the different configurations
   Pink: Non-secure
   Green: NSC (non-secure callable)
   Lighter green: Secure or non-secure or NSC

2.2.1 Default TrustZone security state

When the TrustZone security is activated by the TZEN option bit in the FLASH_OPTR, the default system security state is:

- CPU:
  - Cortex-M33 is in secure state after reset. The boot address must be in secure address (i.e. belonging to the secure part of the memory map).
- Memory map:
  - SAU: is fully secure after reset. Consequently, the memory map is fully secure. Up to 8 SAU configurable regions are available for security attribution.
- Flash:
  - Flash secure regions are defined by watermark user options.
  - Flash interface blocks (i.e. pages) are all Non-Secure after reset (but the watermark configuration supersedes the block-based one in case of overlap).
- SRAMs:
  - All SRAMs are secure after reset. MPCBB (memory protection block based controller) is configured as fully secure.
- External memories:
  - FSMC, OCTOSPI banks are secure after reset. MPCWM (memory protection watermark based controller) is secure
- All peripherals (except GPIOs) are non secure after reset. For TrustZone-aware peripherals, their secure configuration registers are secure.

Note: Refer to Table 2 and Table 3 for a list of Securable and TrustZone-aware peripherals.

- All GPIO are secure after reset.
- Interrupts:
  - NVIC: All interrupts are secure after reset. NVIC is banked for secure and non-secure state.
  - TZIC: All illegal access interrupts are disabled after reset.

2.2.2 TrustZone peripheral classification

When the TrustZone security is active, a peripheral can be either Securable or TrustZone-aware type as defined below:

- Securable: a peripheral is protected by an AHB/APB firewall gate that is controlled from TZSC controller to define security properties.
- TrustZone-aware: a peripheral connected directly to AHB or APB bus and implementing a specific TrustZone behavior such as a subset of registers being secure. TrustZone-aware AHB masters always drive HNONSEC signal according to their security mode (as CM33 core and DMA).

Refer to *Section 5.2.1: GTZC TrustZone system architecture*

Table 2 and *Table 3* summarize the list of Securable and TrustZone aware peripherals within the system.

Table 2. Securable peripherals by TZSC

Bus	Peripheral
AHB3	OCTOSPI1 registers
	FMC registers
AHB 2	SDMMC1
	PKA
	RNG
	HASH
	AES
	ADC
AHB1	ICACHE registers
	TSC
	CRC
APB2	DFSDM1
	SAI2
	SAI1
	TIM17
	TIM16
	TIM15
	USART1
	TIM8
	SPI1
	TIM1
	COMP
	VREFBUF

Table 2. Securable peripherals by TZSC (continued)

Bus	Peripheral
APB1	UCPD1
	USB FS
	FDCAN1
	LPTIM3
	LPTIM2
	I2C4
	LPUART1
	LPTIM1
	OPAMP
	DAC1/DAC2
	CRS
	I2C3
	I2C2
	I2C1
	UART5
	UART4
	USART3
	USART2
	SPI3
	SPI2
	IWDG
	WWDG
	TIM7
	TIM6
	TIM5
	TIM4
	TIM3
	TIM2

Table 3. TrustZone-aware peripherals

Bus	Peripheral
AHB2	GPIOH
	GPIOG
	GPIOF
	GPIOE
	GPIOD
	GPIOC
	GPIOB
	GPIOA
	OTFDEC1(1)
AHB1	GTZC
	EXTIT
	Flash memory
	RCC
	DMAMUX1
	DMA2
	DMA1
APB2	SYSCFG
APB1	PWR
	RTC
	TAMP

1. Always secure when TZEN = 1.

2.3 Memory organization

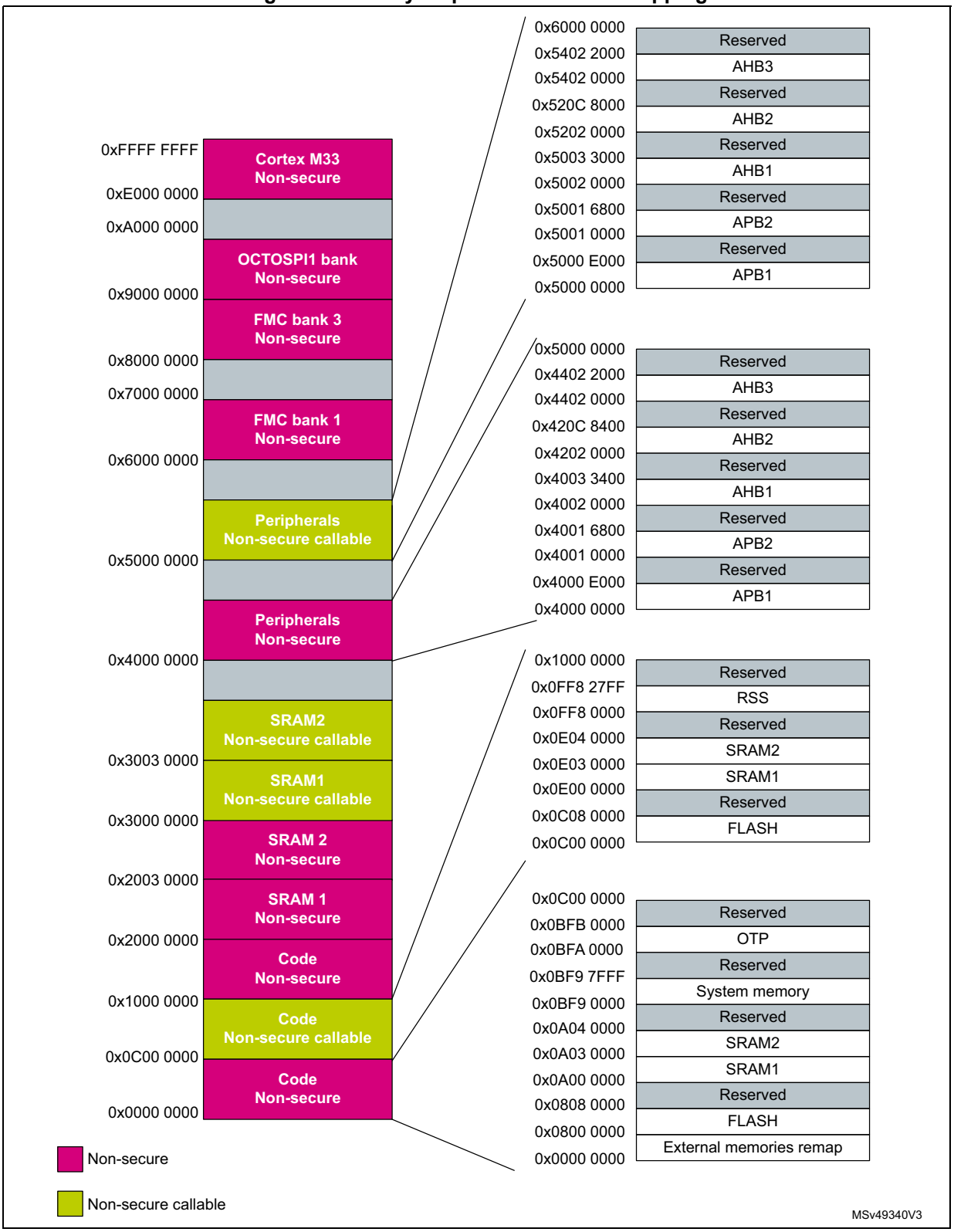
2.3.1 Introduction

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

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

2.3.2 Memory map and register boundary addresses

Figure 2. Memory map based on IDAU mapping



All the memory map areas that are not allocated to 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 4. STM32L552xx and STM32L562xx memory map and peripheral register boundary addresses

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB3	0x5402 1400 - 0x5FFF FFFF	0x4402 1400 - 0x4FFF FFFF	129 MB	Reserved	-
	0x5402 1000 - 0x5402 13FF	0x4402 1000 - 0x4402 13FF	1 KB	OCTOSPI1 registers	*Section 20.7.28: OCTOSPI register map*
	0x5402 0400 - 0x5402 0FFF	0x4402 0400 - 0x4402 0FFF	3 KB	Reserved	-
	0x5402 0000 - 0x5402 03FF	0x4402 0000 - 0x4402 03FF	1 KB	FMC registers	*Section 19.7.8: FMC register map*
AHB2	0x520C 8400 - 0x5401 FFFF	0x420C 8400 - 0x4401 FFFF	32 MB	Reserved	-
	0x520C 8000 - 0x520C 83FF	0x420C 8000 - 0x420C 83FF	1 KB	SDMMC1	*Section 48.9.20: SDMMC register map*
	0x520C 5400 - 0x520C 7FFF	0x420C 5400 - 0x420C 7FFF	11 KB	Reserved	-
	0x520C 5000 - 0x520C 53FF	0x420C 5000 - 0x420C 53FF	1 KB	OTFDEC1	*Section 31.6.15: OTFDEC register map*
	0x520C 4000 - 0x520C 4FFF	0x420C 4000 - 0x420C 4FFF	4 KB	Reserved	-
	0x520C 2000 - 0x520C 3FFF	0x420C 2000 - 0x420C 3FFF	8 KB	PKA	*Section 32.7.5: PKA register map*
	0x520C 0C00 - 0x520C 1FFF	0x420C 0C00 - 0x420C 1FFF	5 KB	Reserved	-
	0x520C 0800 - 0x520C 0BFF	0x420C 0800 - 0x420C 0BFF	1 KB	RNG	*Section 28.7.5: RNG register map*
	0x520C 0400 - 0x520C 07FF	0x420C 0400 - 0x420C 07FF	1 KB	HASH	*Section 30.7.8: HASH register map*
	0x520C 0000 - 0x520C 03FF	0x420C 0000 - 0x420C 03FF	1 KB	AES	*Section 29.7.18: AES register map*
	0x5202 8400 - 0x520B FFFF	0x4202 8400 - 0x420B FFFF	609 KB	Reserved	-
	0x5202 8000 - 0x5202 83FF	0x4202 8000 - 0x4202 83FF	1 KB	ADC	*Section 21.8: ADC register map on page 793*
	0x5202 2000 - 0x5202 7FFF	0x4202 2000 - 0x4202 7FFF	24 KB	Reserved	-
	0x5202 1C00 - 0x5202 1FFF	0x4202 1C00 - 0x4202 1FFF	1 KB	GPIOH	*Section 11.6.13: GPIO register map*
	0x5202 1800 - 0x5202 1BFF	0x4202 1800 - 0x4202 1BFF	1 KB	GPIOG	*Section 11.6.13: GPIO register map*
	0x5202 1400 - 0x5202 17FF	0x4202 1400 - 0x4202 17FF	1 KB	GPIOF	*Section 11.6.13: GPIO register map*

Table 4. STM32L552xx and STM32L562xx memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB2 (continued)	0x5202 1000 - 0x5202 13FF	0x4202 1000 - 0x4202 13FF	1 KB	GPIOE	*Section 11.6.13: GPIO register map*
	0x5202 0C00 - 0x5202 0FFF	0x4202 0C00 - 0x4202 0FFF	1 KB	GPIOD	*Section 11.6.13: GPIO register map*
	0x5202 0800 - 0x5202 0BFF	0x4202 0800 - 0x4202 0BFF	1 KB	GPIOC	*Section 11.6.13: GPIO register map*
	0x5202 0400 - 0x5202 07FF	0x4202 0400 - 0x4202 07FF	1 KB	GPIOB	*Section 11.6.13: GPIO register map*
	0x5202 0000 - 0x5202 03FF	0x4202 0000 - 0x4202 03FF	1 KB	GPIOA	*Section 11.6.13: GPIO register map*
AHB1	0x5003 4000 - 0x5201 FFFF	0x4003 3400 - 0x4201 FFFF	32 MB	Reserved	-
	0x5003 2400 - 0x5003 33FF	0x4003 2400 - 0x4003 33FF	4 KB	GTZC	*Section 5.6.5: GTZC_MPCBB1 register map and reset values* *Section 5.6.6: GTZC_MPCBB2 register map and reset values* *Section 5.7.10: GTZC_TZIC register map and reset values* *Section 5.5.8: GTZC_TZSC register map and reset values*
	0x5003 0800 - 0x5003 23FF	0x4003 0800 - 0x4003 23FF	7 KB	Reserved	-
	0x5003 0400 - 0x5003 07FF	0x4003 0400 - 0x4003 07FF	1 KB	ICache	*Section 7.7.8: ICACHE register map*
	0x5002 F800 - 0x5003 03FF	0x4002 F800 - 0x4003 03FF	3 KB	Reserved	-
	0x5002 F400 - 0x5002 F7FF	0x4002 F400 - 0x4002 F7FF	1 KB	EXTI	*Section 17.6.21: EXTI register map*
	0x5002 4400 - 0x5002 F3FF	0x4002 4400 - 0x4002 F3FF	43 KB	Reserved	-
	0x5002 4000 - 0x5002 43FF	0x4002 4000 - 0x4002 43FF	1 KB	TSC	*Section 27.6.11: TSC register map*
	0x5002 3400 - 0x5002 3FFF	0x4002 3400 - 0x4002 3FFF	3 KB	Reserved	-
	0x5002 3000 - 0x5002 33FF	0x4002 3000 - 0x4002 33FF	1 KB	CRC	*Section 18.4.6: CRC register map*
	0x5002 2400 - 0x5002 2FFF	0x4002 2400 - 0x4002 2FFF	3 KB	Reserved	-
	0x5002 2000 - 0x5002 23FF	0x4002 2000 - 0x4002 23FF	1 KB	Flash registers	*Section 6.9.28: FLASH register map and reset values*

Table 4. STM32L552xx and STM32L562xx memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB1 (continued)	0x5002 1400 - 0x5002 1FFF	0x4002 1400 - 0x4002 1FFF	3 KB	Reserved	-
	0X5002 1000 - 0x5002 13FF	0X4002 1000 - 0x4002 13FF	1 KB	RCC	*Section 9.8.42: RCC register map*
	0x5002 0C00 - 0x5002 0FFF	0x4002 0C00 - 0x4002 0FFF	1 KB	Reserved	-
	0x5002 0800 - 0x5002 0BFF	0x4002 0800 - 0x4002 0BFF	1 KB	DMAMUX1	*Section 15.6.7: DMAMUX register map*
	0x5002 0400 - 0x5002 07FF	0x4002 0400 - 0x4002 07FF	1 KB	DMA2	*Section 14.6.8: DMA register map*
	0x5002 0000 - 0x5002 03FF	0x4002 0000 - 0x4002 03FF	1 KB	DMA1	*Section 14.6.8: DMA register map*
APB2	0x5001 6800 - 0x5001 FFFF	0x4001 6800 - 0x4001 FFFF	38 KB	Reserved	-
	0x5001 6000 - 0x5001 67FF	0x4001 6000 - 0x4001 67FF	2 KB	DFSDM1	*Section 26.8.16: DFSDM register map*
	0x5001 5C00 - 0x5001 5FFF	0x4001 5C00 - 0x4001 5FFF	1 KB	Reserved	-
	0x5001 5800 - 0x5001 5BFF	0x4001 5800 - 0x4001 5BFF	1 KB	SAI2	*Section 47.6.20: SAI register map*
	0x5001 5400 - 0x5001 57FF	0x4001 5400 - 0x4001 57FF	1 KB	SAI1	*Section 47.6.20: SAI register map*
	0x5001 4C00 - 0x5001 53FF	0x4001 4C00 - 0x4001 53FF	2 KB	Reserved	-
	0x5001 4800 - 0x5001 4BFF	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17	*Section 35.6.21: TIM16/TIM17 register map*
	0x5001 4400 - 0x5001 47FF	0x4001 4400 - 0x4001 47FF	1 KB	TIM16	*Section 35.6.21: TIM16/TIM17 register map*
	0x5001 4000 - 0x5001 43FF	0x4001 4000 - 0x4001 43FF	1 KB	TIM15	*Section 35.5.21: TIM15 register map*
	0x5001 3C00 - 0x5001 3FFF	0x4001 3C00 - 0x4001 3FFF	1 KB	Reserved	-
	0x5001 3800 - 0x5001 3BFF	0x4001 3800 - 0x4001 3BFF	1 KB	USART1	*Section 44.8.15: USART register map*
	0x5001 3400 - 0x5001 37FF	0x4001 3400 - 0x4001 37FF	1 KB	TIM8	*Section 33.4.33: TIM8 register map*
	0x5001 3000 - 0x5001 33FF	0x4001 3000 - 0x4001 33FF	1 KB	SPI1	*Section 46.6.8: SPI register map*
	0x5001 2C00 - 0x5001 2FFF	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1	*Section 33.4.32: TIM1 register map*

Table 4. STM32L552xx and STM32L562xx memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
APB2 (continued)	0x5001 0400 - 0x5001 2BFF	0x4001 0400 - 0x4001 2BFF	10 KB	Reserved	-
	0x5001 0200 - 0x5001 03FF	0x4001 0200 - 0x4001 03FF	1 KB	COMP	*Section 24.6.3: COMP register map*
	0x5001 0100 - 0x5001 01FF	0x4001 0100 - 0x4001 01FF	1 KB	VREFBUF	*Section 23.4.3: VREFBUF register map*
	0x5001 0000 - 0x5001 002F	0x4001 0000 - 0x4001 002F	1 KB	SYSCFG	*Section 12.3.12: SYSCFG register map*
APB1	0x5000 E000 - 0x5000 FFFF	0x4000 E000 - 0x4000 FFFF	8 KB	Reserved	-
	0x5000 DC00 - 0x5000 DFFF	0x4000 DC00 - 0x4000 DFFF	1 KB	UCPD1	*Section 51.7.16: UCPD register map*
	0x5000 D800 - 0x5000 DBFF	0x4000 D800 - 0x4000 DBFF	1 KB	USB SRAM	*Section 50.6.3: USB register map*
	0x5000 D400 - 0x5000 D7FF	0x4000 D400 - 0x4000 D7FF	1 KB	USB FS	*Section 50.6.3: USB register map*
	0x5000 B000 - 0x5000 D3FF	0x4000 B000 - 0x4000 D3FF	9 KB	Reserved	-
	0x5000 AC00 - 0x5000 AFFF	0x4000 AC00 - 0x4000 AFFF	1 KB	FDCAN RAM	*Section 49.4.38: FDCAN register map*
	0x5000 A800 - 0x5000 ABFF	0x4000 A800 - 0x4000 ABFF	1 KB	Reserved	-
	0x5000 A400 - 0x5000 A7FF	0x4000 A400 - 0x4000 A7FF	1 KB	FDCAN1	*Section 49.4.38: FDCAN register map*
	0x5000 9C00 - 0x5000 A3FF	0x4000 9C00 - 0x4000 A3FF	2 KB	Reserved	-
	0x5000 9800 - 0x5000 9BFF	0x4000 9800 - 0x4000 9BFF	1 KB	LPTIM3	*Section 37.7.13: LPTIM register map*
	0x5000 9400 - 0x5000 97FF	0x4000 9400 - 0x4000 97FF	1 KB	LPTIM2	*Section 37.7.13: LPTIM register map*
	0x5000 8800 - 0x5000 93FF	0x4000 8800 - 0x4000 93FF	3 KB	Reserved	-
	0x5000 8400 - 0x5000 87FF	0x4000 8400 - 0x4000 87FF	1 KB	I2C4	*Section 43.7.16: I2C register map*
	0x5000 8000 - 0x5000 83FF	0x4000 8000 - 0x4000 83FF	1 KB	LPUART1	-
	0x5000 7C00 - 0x5000 7FFF	0x4000 7C00 - 0x4000 7FFF	1 KB	LPTIM1	*Section 37.7.13: LPTIM register map*
	0x5000 7800 - 0x5000 7BFF	0x4000 7800 - 0x4000 7BFF	1 KB	OPAMP	*Section 25.5: OPAMP registers*

Table 4. STM32L552xx and STM32L562xx memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1 (continued)	0x5000 7400 - 0x5000 77FF	0x4000 7400 - 0x4000 77FF	1 KB	DAC	*Section 22.7.21: DAC register map*
	0x5000 7000 - 0x5000 73FF	0x4000 7000 - 0x4000 73FF	1 KB	PWR	*Section 8.6.26: PWR register map and reset values*
	0x5000 6400 - 0x5000 6FFF	0x4000 6400 - 0x4000 6FFF	3 KB	Reserved	-
	0x5000 6000 - 0x5000 63FF	0x4000 6000 - 0x4000 63FF	1 KB	CRS	*Section 10.7.5: CRS register map*
	0x5000 5C00 - 0x5000 5FFF	0x4000 5C00 - 0x4000 5FFF	1 KB	I2C3	*Section 43.7.16: I2C register map*
	0x5000 5800 - 0x5000 5BFF	0x4000 5800 - 0x4000 5BFF	1 KB	I2C2	*Section 43.7.16: I2C register map*
	0x5000 5400 - 0x5000 57FF	0x4000 5400 - 0x4000 57FF	1 KB	I2C1	*Section 43.7.16: I2C register map*
	0x5000 5000 - 0x5000 53FF	0x4000 5000 - 0x4000 53FF	1 KB	UART5	*Section 44.8.15: USART register map*
	0x5000 4C00 - 0x5000 4FFF	0x4000 4C00 - 0x4000 4FFF	1 KB	UART4	*Section 44.8.15: USART register map*
	0x5000 4800 - 0x5000 4BFF	0x4000 4800 - 0x4000 4BFF	1 KB	USART3	*Section 44.8.15: USART register map*
	0x5000 4400 - 0x5000 47FF	0x4000 4400 - 0x4000 47FF	1 KB	USART2	*Section 44.8.15: USART register map*
	0x5000 4000 - 0x5000 43FF	0x4000 4000 - 0x4000 43FF	1 KB	Reserved	-
	0x5000 3C00 - 0x5000 3FFF	0x4000 3C00 - 0x4000 3FFF	1 KB	SPI3	*Section 46.6.8: SPI register map*
	0x5000 3800 - 0x5000 3BFF	0x4000 3800 - 0x4000 3BFF	1 KB	SPI2	*Section 46.6.8: SPI register map*
	0x5000 3400 - 0x5000 37FF	0x4000 3400 - 0x4000 37FF	1 KB	TAMP	*Section 42.6.19: TAMP register map*
	0x5000 3000 - 0x5000 33FF	0x4000 3000 - 0x4000 33FF	1 KB	IWDG	*Section 39.4.6: IWDG register map*
	0x5000 2C00 - 0x5000 2FFF	0x4000 2C00 - 0x4000 2FFF	1 KB	WWDG	*Section 40.5.4: WWDG register map*
	0x5000 2800 - 0x5000 2BFF	0x4000 2800 - 0x4000 2BFF	1 KB	RTC	*Section 41.6.24: RTC register map*
	0x5000 1800 - 0x5000 27FF	0x4000 1800 - 0x4000 27FF	4 KB	Reserved	-
	0x5000 1400 - 0x5000 17FF	0x4000 1400 - 0x4000 17FF	1 KB	TIM7	*Section 36.4.9: TIMx register map*

Table 4. STM32L552xx and STM32L562xx memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1 (continued)	0x5000 1000 - 0x5000 13FF	0x4000 1000 - 0x4000 13FF	1 KB	TIM6	*Section 36.4.9: TIMx register map*
	0x5000 0C00 - 0x5000 0FFF	0x4000 0C00 - 0x4000 0FFF	1 KB	TIM5	*Section 34.4.26: TIMx register map*
	0x5000 0800 - 0x5000 0BFF	0x4000 0800 - 0x4000 0BFF	1 KB	TIM4	*Section 34.4.26: TIMx register map*
	0x5000 0400 - 0x5000 07FF	0x4000 0400 - 0x4000 07FF	1 KB	TIM3	*Section 34.4.26: TIMx register map*
	0x5000 0000 - 0x5000 03FF	0x4000 0000 - 0x4000 03FF	1 KB	TIM2	*Section 34.4.26: TIMx register map*

2.4 Embedded SRAM

The STM32L552xx and STM32L562xx devices feature up to 256 Kbytes SRAM:

- 192 Kbytes SRAM1
- 64 Kbytes 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 and SRAM2 through the system bus or through the C-bus depending on the selected address.

Either 64 Kbytes or upper 4 Kbytes of SRAM2 can be retained in Standby mode.

When the TrustZone security is enabled, all SRAMs are secure after reset. The SRAM can be programmed as non-secure with a block granularity, using MPCBB (memory protection controller block configuration based) in GTZC controller. The granularity of SRAM secure/non-secure block-based is a page of 256 bytes.

2.4.1 SRAM2 parity check

The user can enable the SRAM2 parity check using the option bit SRAM2_PE in the OPTR user option register (refer to *Section 6.4.1: 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 safety standards.

The parity bits are computed and stored when writing into the SRAM2. 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 with the SPL control bit in the *SYSCFG configuration register 2 (SYSCFG_CFGR2)*. The SRAM2 Parity Error flag (SPF) is available in the *SYSCFG configuration register 2 (SYSCFG_CFGR2)*.

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

2.4.2 SRAM2 Write protection

The SRAM2 can be write protected with a page granularity of 1 Kbyte.

Table 5. SRAM2 organization

Page number	Start address	End address
Page 0	0x2003 0000	0x2003 03FF
Page 1	0x2003 0400	0x2003 07FF
Page 2	0x2003 0800	0x2003 0BFF
Page 3	0x2003 0C00	0x2003 0FFF
Page 4	0x2003 1000	0x2003 13FF
Page 5	0x2003 1400	0x2003 17FF
Page 6	0x2003 1800	0x2003 1BFF
Page 7	0x2003 1C00	0x2003 1FFF
Page 8	0x2003 2000	0x2003 23FF
Page 9	0x2003 2400	0x2003 27FF
Page 10	0x2003 2800	0x2003 2BFF
Page 11	0x2003 2C00	0x2003 2FFF
Page 12	0x2003 3000	0x2003 33FF
Page 13	0x2003 3400	0x2003 37FF
Page 14	0x2003 3800	0x2003 3BFF
Page 15	0x2003 3C00	0x2003 3FFF
Page 16	0x2003 4000	0x2003 43FF
Page 17	0x2003 4400	0x2003 47FF
Page 18	0x2003 4800	0x2003 4BFF
Page 19	0x2003 4C00	0x2003 4FFF
Page 20	0x2003 5000	0x2003 53FF
Page 21	0x2003 5400	0x2003 57FF
Page 22	0x2003 5800	0x2003 5BFF
Page 23	0x2003 5C00	0x2003 5FFF
Page 24	0x2003 6000	0x2003 63FF
Page 25	0x2003 6400	0x2003 67FF
Page 26	0x2003 6800	0x2003 6BFF
Page 27	0x2003 6C00	0x2003 6FFF
Page 28	0x2003 7000	0x2003 73FF

Table 5. SRAM2 organization (continued)

Page number	Start address	End address
Page 29	0x2003 7400	0x2003 77FF
Page 30	0x2003 7800	0x2003 7BFF
Page 31	0x2003 7C00	0x2003 7FFF
Page 32	0x2003 8000	0x2003 83FF
Page 33	0x2003 8400	0x2003 87FF
Page 34	0x2003 8800	0x2003 8BFF
Page 35	0x2003 8C00	0x2003 8FFF
Page 36	0x2003 9000	0x2003 93FF
Page 37	0x2003 9400	0x2003 97FF
Page 38	0x2003 9800	0x2003 9BFF
Page 39	0x2003 9C00	0x2003 9FFF
Page 40	0x2003 A000	0x2003 A3FF
Page 41	0x2003 A400	0x2003 A7FF
Page 42	0x2003 A800	0x2003 ABFF
Page 43	0x2003 AC00	0x2003 AFFF
Page 44	0x2003 B000	0x2003 B3FF
Page 45	0x2003 B400	0x2003 B7FF
Page 46	0x2003 B800	0x2003 BBFF
Page 47	0x2003 BC00	0x2003 BFFF
Page 48	0x2003 C000	0x2003 C3FF
Page 49	0x2003 C400	0x2003 C7FF
Page 50	0x2003 C800	0x2003 CBFF
Page 51	0x2003 CC00	0x2003 CFFF
Page 52	0x2003 D000	0x2003 D3FF
Page 53	0x2003 D400	0x2003 D7FF
Page 54	0x2003 D800	0x2003 DBFF
Page 55	0x2003 DC00	0x2003 DFFF
Page 56	0x2003 E000	0x2003 E3FF
Page 57	0x2003 E400	0x2003 E7FF
Page 58	0x2003 E800	0x2003 EBFF
Page 59	0x2003 EC00	0x2003 EFFF
Page 60	0x2003 F000	0x2003 F3FF
Page 61	0x2003 F400	0x2003 F7FF
Page 62	0x2003 F800	0x2003 FBFF
Page 63	0x2003 FC00	0x2003 FFFF

The write protection can be enabled in *SYSCFG SRAM2 write protection register (SYSCFG_SWPR)* in the SYSCFG block. This is a register with write ‘1’ once mechanism, which means that writing ‘1’ on a bit will setup the write protection for that page of SRAM and it can be removed/cleared by a system reset only.

2.4.3 SRAM2 Read protection

The SRAM2 is protected with the Read protection (RDP). Refer to *Section 6.7.2: Readout protection (RDP)* for more details.

2.4.4 SRAM2 Erase

The SRAM2 can be erased with a system reset using the option bit SRAM2_RST in the OPTR user option register (refer to *Section 6.4.1: Option bytes description*).

The SRAM2 erase can also be requested by software by setting the bit SRAM2ER in the *SYSCFG SRAM2 control and status register (SYSCFG_SCSR)*.

The SRAM2 is also erased by a Backup domain reset.

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 plus security access control features.Refer to *Section 6: Embedded Flash memory (FLASH)* for more details.

3 Boot configuration

At startup, a BOOT0 pin, nBOOT0 and NSBOOTADDx[24:0] / SECBOOTADD0[24:0] option bytes are used to select the boot memory address which includes:

- Boot from any address in user Flash
- Boot from system memory bootloader
- Boot from any address in embedded SRAM
- Boot from Root Security service (RSS)

The BOOT0 value may come from the PH3-BOOT0 pin or from an option bit depending on the value of a user option bit to free the GPIO pad if needed.

Refer to *Table 6* and *Table 7* for boot modes when TrustZone is disabled and enabled respectively.

Table 6. Boot modes when TrustZone is disabled (TZEN=0)

nBOOT0 FLASH_ OPTR[27]	BOOT0 pin PH3	nSWBOOT0 FLASH_ OPTR[26]	Boot address option-bytes selection	Boot area	ST programmed default value
-	0	1	NSBOOTADD0[24:0]	Boot address defined by user option bytes NSBOOTADD0[24:0]	Flash: 0x0800 0000
-	1	1	NSBOOTADD1[24:0]	Boot address defined by user option bytes NSBOOTADD1[24:0]	System bootloader: 0x0BF9 0000
1	-	0	NSBOOTADD0[24:0]	Boot address defined by user option bytes NSBOOTADD0[24:0]	Flash: 0x0800 0000
0	-	0	NSBOOTADD1[24:0]	Boot address defined by user option bytes NSBOOTADD1[24:0]	System bootloader: 0x0BF9 0000

When TrustZone is enabled by setting the TZEN option bit, the boot space must be in secure area. The SECBOOTADD0[24:0] option bytes are used to select the boot secure memory address.

A unique boot entry option can be selected by setting the BOOT_LOCK option bit. All other boot options are ignored.

Table 7. Boot modes when TrustZone is enabled (TZEN=1)

BOOT_LOCK	nBOOT0 FLASH_OPTR[27]	BOOT0 pin PH3	nSWBOOT0 FLASH_OPTR[26]	RSS command	Boot address option-bytes selection	Boot area	ST programmed default value
0	-	0	1	0	SECBOOTADD0[24:0]	Secure boot address defined by user option bytes SECBOOTADD0[24:0]	Flash: 0x0C00 0000
	-	1	1	0	N/A	RSS: 0x0FF8 0000	RSS: 0x0FF8 0000
	1	-	0	0	SECBOOTADD0[24:0]	Secure boot address defined by user option bytes SECBOOTADD0[24:0]	Flash: 0x0C00 0000
	0	-	0	0	N/A	RSS: RSS: 0x0FF8 0000	RSS: 0x0FF8 0000
	-	-	-	≠ 0	N/A	RSS: RSS: 0x0FF8 0000	RSS: 0x0FF8 0000
1	-	-	-	-	SECBOOTADD0[24:0]	Secure boot address defined by user option bytes SECBOOTADD0[24:0]	Flash: 0x0C00 0000

The boot address option bytes enables the possibility to program any boot memory address. However, the allowed address space depends on Flash read protection RDP level.

If the programmed boot memory address is out of the allowed memory mapped area when RDP level is 0.5 or more, the default boot fetch address is forced to:

- 0x0800 0000 (when TZEN = 0)
- RSS (when TZEN = 1)

Refer to the *Table 8*.

Table 8. Boot space versus RDP protection

RDP	TZEN = 1	TZEN = 0
0	Any boot address	Any boot address
0.5	Boot address only in: RSS: 0x0FF80000 or secure Flash: 0x0C000000 - 0x0C07 FFFF Otherwise the boot address is forced to RSS	N/A
1		Any boot address
2		Boot address only in Flash 0x0800 0000 - 0x0807 FFFF Otherwise the forced boot address is: 0x0800 0000(1)

1. In RDP level 2, the boot is done from the address programmed in NSBOOTADD0 or NSBOOTADD1 depending on the boot configuration before setting the RDP level 2 and if the programmed address is within the user Flash memory.
If the programmed NSBOOTADD0 or NSBOOTADD1 is not a valid address, the boot if forced at 0x800 0000.

The BOOT0 value (either coming from the pin or the option bit) is latched upon reset release. It is up to the user to set nBOOT0 or BOOT0 values to select the required boot mode.

The BOOT0 pin or user option bit (depending on the nSWBOOT0 bit value in the FLASH_OPTR register) is also re-sampled when exiting from Standby mode. Consequently, they must be kept in the required Boot mode configuration in Standby mode.After startup delay, the selection of the boot area is done before releasing the processor reset.

PH3/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.

Embedded bootloader and RSS

The bootloader is located in the system memory. It is used to reprogram the Flash memory by using USART, I2C, SPI, FDCAN or USB FS in device mode through the DFU (device firmware upgrade). It is programmed by ST during production. Refer to *AN2606, STM32 microcontroller system memory boot mode*.

The root secure services (RSS) are embedded in a Flash memory area named secure information block, programmed during ST production.

The RSS enables for example the secure firmware installation (SFI) thanks to the RSS extension firmware (RSSe SFI).

This feature allows the customers to protect the confidentiality of the firmware to be provisioned into the STM32 device when the production is subcontracted to a third party.

The RSS is available on all devices, after enabling the TrustZone through the TZEN option bit. Refer to *AN5428, STM32L5 Series microcontroller system memory RSS services*.

4 System security

4.1 Introduction

The STM32L552xx and STM32L562xx have been designed with a comprehensive set of security features, some of which being based on standard Arm TrustZone® technology. These security features should simplify the process of evaluating IoT devices against security standards. They also significantly reduce the cost and complexity of software development for OEMs and third party developers by facilitating the re-use, improving the interoperability, and minimizing the API fragmentation.

This section explains the different security features available on the STM32L552xx and STM32L562xx devices.

4.2 Key security features

- Resource isolation using Armv8-M mainline security extension of Cortex®-M33, extended to securable I/Os, memories and peripherals
- Secure firmware installation (SFI) with device unique cryptographic key pair
  - Leveraging the on-chip immutable bootloader that supports the download of image through USART, USB, I2C, SPI, FDCAN and JTAG
- Enabled for secure boot thanks to unique boot entry feature and hide protect area (HDP) mechanism.
- Secure storage, featuring:
  - Non-volatile on-chip secure storage, protected with secure & HDP areas
  - Battery-powered volatile secure storage, automatically erased in case of tamper
  - Write-only key registers in AES engine
  - STM32L552xx and STM32L562xx Unique ID (96 bits)
- General purpose cryptographic acceleration (AES and PKA only in STM32L562xx)
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM and CCM chaining modes
  - HASH processor, supporting MD5/SHA-1 checksums and SHA-2 secure hash
  - Public key accelerator (PKA) for RSA/DH (up to 3136 bits) and ECC (up to 640 bits)
  - True random number generator (RNG), NIST SP800-90B pre-certified
- On-the-fly decryption of encrypted image stored on external Flash memory connected through OCTOSPI (STM32L562xx only)
  - Almost-zero latency with standard NOR Flash memories
  - Can be used to encrypt the image using device unique secret keys
  - Automatic key erase in case of tamper
- Flexible lifecycle scheme with readout protection (RDP), including support for product decommissioning (auto-erase)
  - Debug protection, depending on the readout protection level
- Protected firmware distribution scheme, using TrustZone®, on-the-fly decryption and RDP level 0.5
- Active tamper and protection against temperature, voltage and frequency attacks
  - 3 active inputs, 1 active output tamper pin available in all power modes

4.3 Secure install

Secure firmware install (SFI) is an immutable secure service embedded by STMicroelectronics in STM32L552xx and STM32L562xx devices. It allows secure and counted installation of OEM firmware in untrusted production environment (such as OEM contract manufacturer).

The confidentiality of the installed images written either in internal Flash memory or encrypted in external Flash memory is also protected, using AES.

The SFI native service leverages the following hardware security features:

- Secure boot (see *Section 4.4*)
- Resource isolation using TrustZone® (see *Section 4.6*)
- Temporal isolation using hide protection (see *Section 4.7.1*)
- Secure execution (see *Section 4.8*)
- Secure storage, with associated cryptographic engines (see *Section 4.9* and *Section 4.10*)

Further information can be found in application note AN4992 - Overview secure firmware install (SFI) available on *www.st.com*.

4.4 Secure boot

4.4.1 Introduction

Secure boot is an immutable code that is always executed after a system reset. As a root of trust, this code checks the STM32L552xx and STM32L562xx static protections and activates available STM32L552xx and STM32L562xx runtime protections, reducing the risk that invalid or malicious code runs on the platform. As root of trust, secure boot also checks integrity and authenticity of the next level firmware before executing it.

The actual functions of secure boot depend on availability of TrustZone® features, and the firmware stored in the device. However it would typically initializes secure storage, and install on-the-fly decryption keys in OTFDEC to be able to use encrypted firmware stored in external Flash memory.

The STM32L552xx and STM32L562xx Trusted Firmware-M (TFM) application, supported by the STM32 ecosystem, provides a root of trust solution including secure boot functions. For more information, refer to user manual UM2671 - *Getting started with STM32CubeL5 TFM application* available from *www.st.com*.

In the STM32L552xx and STM32L562xx devices, the secure boot takes benefit of hardware security features such as:

- Resource isolation using TrustZone® (see *Section 4.6*)
- Temporal isolation using hide protection (see *Section 4.7.1*)
- Secure execution (see *Section 4.8*)
- Secure install and update (see *Section 4.3* and *Section 4.5*)
- Secure storage, with associated cryptographic engines if available (see *Section 4.9* and *Section 4.10*)

This section describes the features specifically designed for secure boot.

4.4.2 Unique boot entry and BOOT_LOCK

When TrustZone® is activated (TZEN=1) and BOOT_LOCK secure option bit is *cleared* the application selects a boot entry point located either in system Flash memory (see *Section 4.4.3*) or in secure user Flash memory, at the address defined by SECBOOTADD0 option bytes.

When TrustZone® is activated (TZEN=1) and BOOT_LOCK secure option bit is *set* the device unique boot entry is the secure address defined by SECBOOTADD0 option bytes. These option bytes cannot be modified by the application anymore.

Note: As long as it is cleared, the BOOT_LOCK option byte can be set without any constraint. But once set the BOOT_LOCK option bit cannot be cleared.

For more information on STM32L552xx and STM32L562xx boot mechanisms, refer to *Section 3: Boot configuration*.

4.4.3 Immutable root of trust in system Flash memory

The first usage of the immutable root of trust code stored in STM32L552xx and STM32L562xx system Flash memory is to perform secure firmware install (SFI), allowing secure and counted installation of OEM firmware in untrusted production environment (such as OEM contract manufacturer). See *Section 4.4.2* for more details.

STMicroelectronics immutable code also includes secure runtime services that can be called at runtime when secure application sets the SYSCFG_RSSCMDR register to a non-null value before triggering a system reset. This runtime feature is deactivated when BOOT_LOCK secure option bit is set.

4.5 Secure update

Secure firmware update is a secure service that runs after a secure boot. Its actual functions depend on availability of TrustZone® features, and the firmware stored in the device.

The STM32L552xx and STM32L562xx Trusted Firmware-M (TFM) application, supported by the STM32 ecosystem, allows the update of the microcontroller built-in program with new firmware versions, adding new features and correcting potential issues. The update process is performed in a secure way to prevent unauthorized updates and access to confidential on-device data.

A firmware update can be done either on a single firmware image including both secure and non-secure parts, or on the secure (respectively non-secure) part of the firmware image, independently.

In the STM32L552xx and STM32L562xx devices, the secure update application leverages the same hardware security as the firmware install described in *Section 4.3*.

For more information, refer to user manual UM2671 - Getting started with STM32CubeL5 TFM application available on *www.st.com*.

4.6 Resource isolation using TrustZone®

4.6.1 Introduction

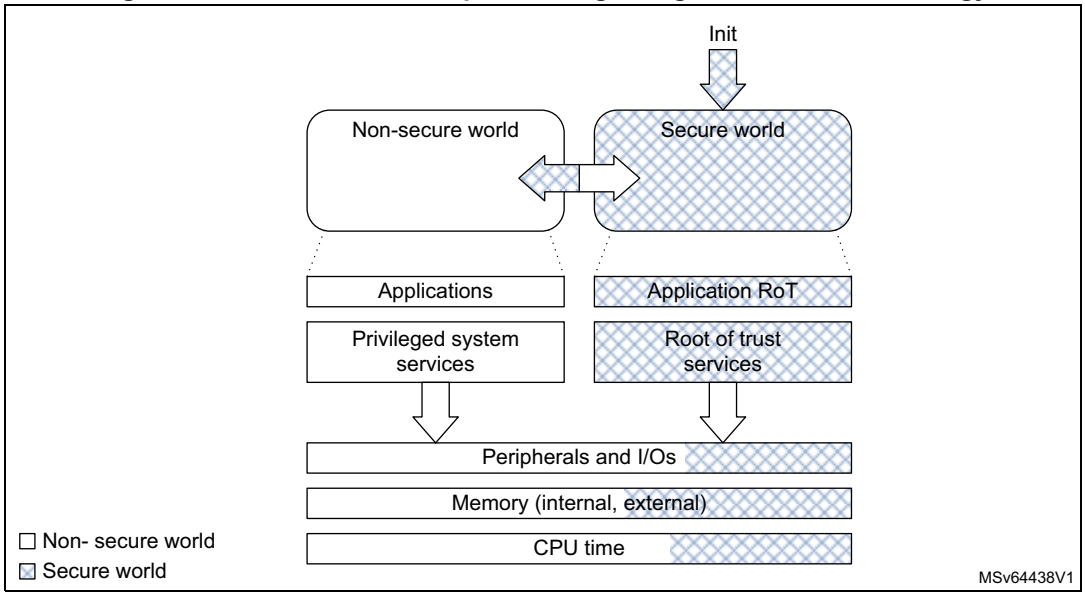
In the STM32L552xx and STM32L562xx devices, the hardware and software resources can be partitioned so that they exist either in the secure world or in the non-secure world, as shown on *Figure 3*.

Note: The initial partitioning of the platform is under the responsibility of the secure firmware executed after reset of the device.

Thanks to this resource isolation technology, the secure world can be used to protect critical code against intentional or unintentional tampering from the more exposed code running in the non-secure world.

Note: Typically secure code is small and rarely modified, while non-secure code is more exposed, and prone to firmware updates.

Figure 3. Secure/non-secure partitioning using TrustZone® technology



4.6.2 TrustZone® security architecture

Armv8-M TrustZone® technology is a comprehensive hardware architecture that proposes to developers a comprehensive, holistic protection across the entire processor and system. In this device TrustZone® hardware features include:

- The Armv8-M mainline security extension of Cortex®-M33, enabling a new processor secure state, with its associated secure interrupts
- The dynamic allocation of memory and peripherals to TrustZone® using eight security attribution unit (SAU) regions of Cortex®-M33
- A global TrustZone® framework (GTZC), extending TrustZone® protection against transactions coming from other masters in the system than the Cortex®-M33.
- TrustZone®-aware embedded Flash memory and peripherals

Note: TrustZone® security is activated by the TZEN option bit in the FLASH_OPTR register

Each of the elements above are described in the following subsections.

4.6.3 Armv8-M security extension of Cortex®-M33

The Arm security extension of the Cortex®-M33 is an evolution, not a revolution. It is using the programmer's model you find in earlier Cortex® M subfamilies like Cortex®-M4. Indeed, Armv8-M is architecturally similar to Armv7-M, using the same 32-bit architecture, the same memory mapped resources protected with an MPU, and it also uses the Nested Vectored Interrupt Controller (NVIC).

Armv8-M TrustZone® implementation in STM32L552xx and STM32L562xx is composed of the following features:

- A new processor state, with almost no additional code/cycle overhead as opposed to Armv8-A TrustZone® that uses a dedicated exception routine for triggering a secure/non-secure world change
- Two memory map views of a shared 4 Gbytes address space
- A low interrupt latency for both secure and non-secure domains. It also includes a new interrupt configuration for security grouping and priority setting.
- Separated exception vector tables for the secure and non-secure exceptions
- Micro-coded context preservation
- Banking of specific registers across secure/non-secure states, including stack pointers with stack-limit checkers
- Banking of following Cortex®-M33 programmable components (two separate units for secure and non-secure):
  - SysTick timer
  - MPU configuration registers (eight MPU regions in secure, eight in non-secure)
  - Some of the system control block (SCB) registers
- New system exception (SecureFault) for handling of security violations
- Configurable debug support, as defined in *Section 4.12: Access controlled debug*

For more information please refer to Cortex®-M33 programming manual (PM0264).

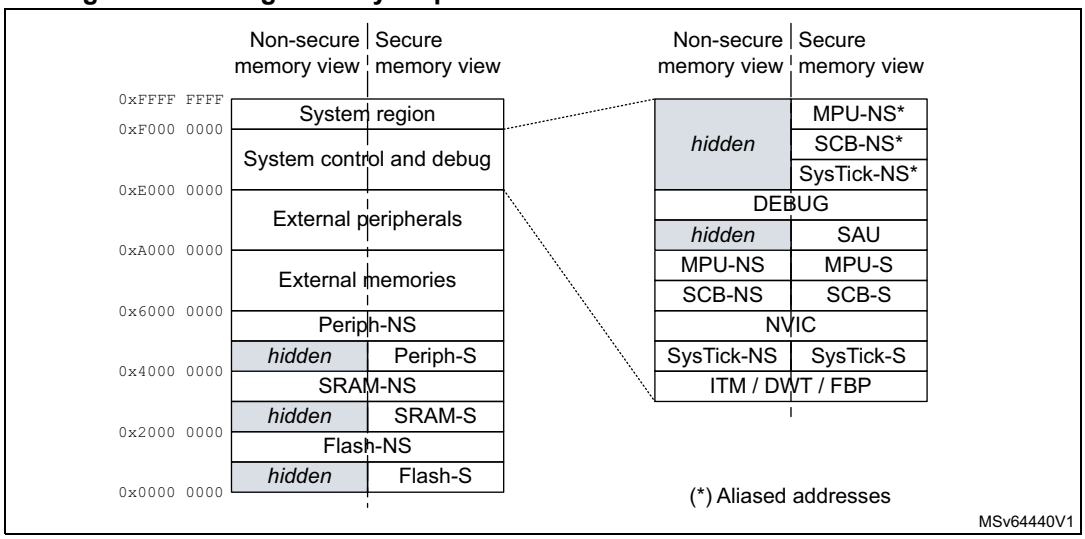
4.6.4 Memory and peripheral allocation using IDAU/SAU

Security attributes

As illustrated on *Figure 4*, Armv8-M non-secure memory view is similar to Armv7-M (that can be found in Cortex® M4), with the difference that secure memory is hidden. The secure memory view shows Flash memory, SRAM and peripherals that are only accessible while the Cortex® processor executes in Secure state.

Note: Figure 4 shows the 32-bit address space viewed after SAU has been configured by the secure code.

Figure 4. Sharing memory map between CPU in secure and non-secure state

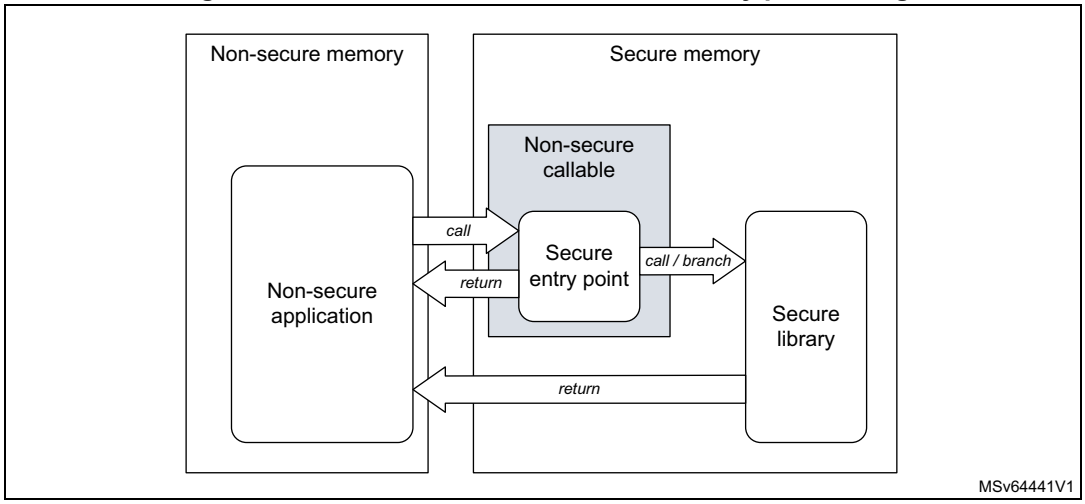


The Cortex® processor state (and associated rights) depends on the security attribute assigned to the memory region where it is executing. More specifically:

- A processor in a non-secure state only executes from *non-secure (NS)* program memory, while a processor in a secure state only executes from *secure (S)* program memory.
- While running in secure state the processor can access data from both S and NS memories. Running in non-secure state the CPU is limited to non-secure memories.

In order to manage transitions to secure world, developers must create *non-secure callable (NSC)* regions that contain valid entry points to the secure libraries. The first instruction in these entry points must be the new **`secure gate`** (SG) instruction, used by non-secure code to call a secure function. It is illustrated on *Figure 5*.

Figure 5. Secure world transition and memory partitioning



Programming security attributes

In Cortex®-M33 the static implementation defined attribution unit (IDAU) works in conjunction with the programmable security attribution unit (SAU) to assign a specific security attribute (S, NS or NSC) to a specific address, as shown on *Table 9*.

Table 9. Configuring security attributes with IDAU and SAU

IDAU security attribution	SAU security attribution(1)	Final security attribution
Non-secure	Secure	Secure
	Secure-NSC	Secure-NSC
	Non-secure	Non-secure
Secure-NSC	Secure	Secure
	Non-secure	Secure-NSC

1. Defined regions are aligned to 32-byte boundaries.

The SAU can only be configured by the Cortex®-M33 in the secure privileged state. When TrustZone® is enabled, the SAU defaults all addresses as secure (S). A secure boot application can then program SAU to create NSC or NS regions, as shown in *Table 9*.

The SAU/IDAU settings are applicable to only the Cortex®-M33. The other masters like DMA are not affected by those policies.

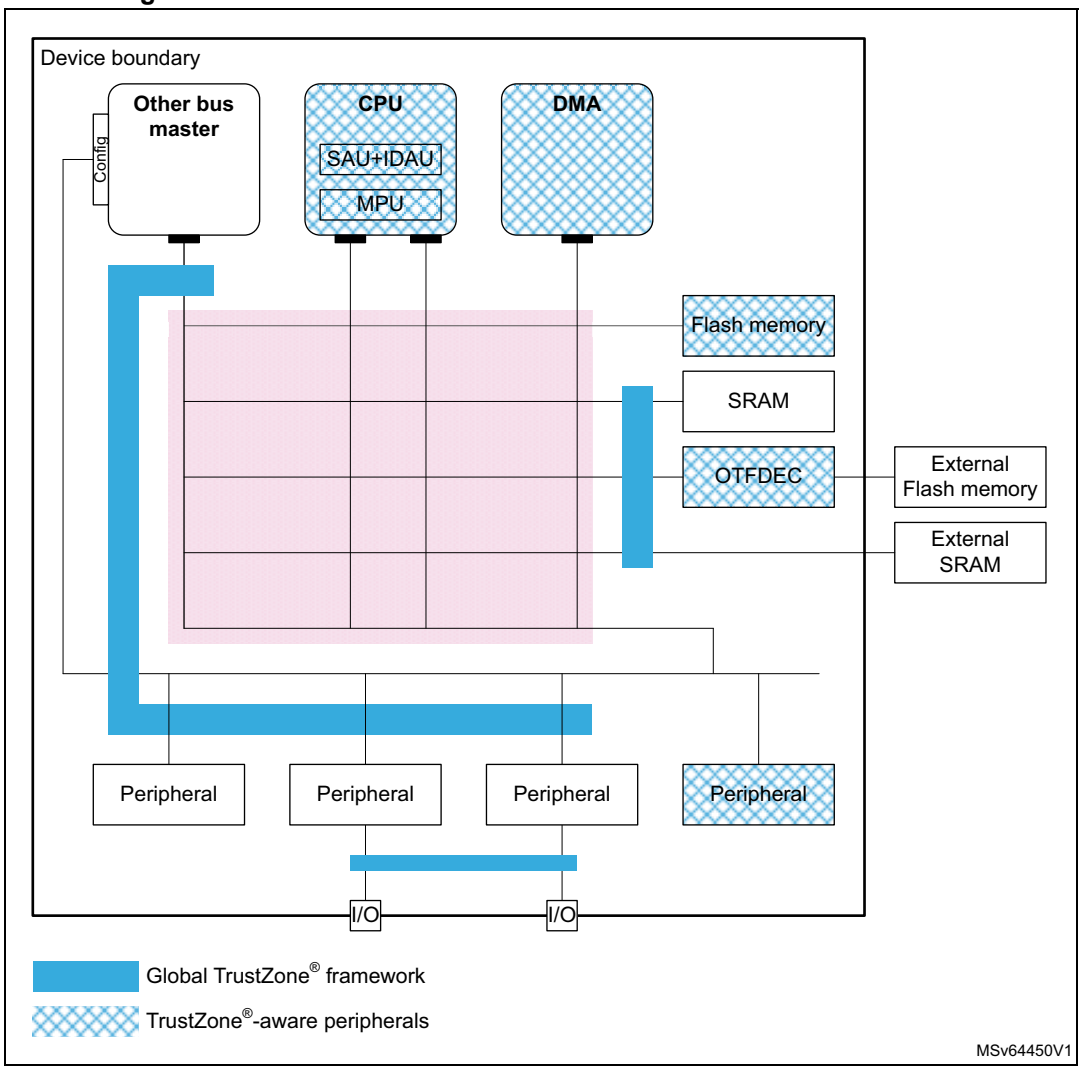
For more information on memory security attribution using IDAU/SAU on STM32L552xx and STM32L562xx, please refer to AN5347 on STMicroelectronics website.

4.6.5 Memory and peripheral allocation using GTZC

Global Trustzone® framework architecture

On top of Armv8-M TrustZone® security extension in Cortex®-M33, the STM32L552xx and STM32L562xx devices come with complementary security features that reinforce in a flexible way the isolation between the secure and the non-secure worlds. Unlike the SAU/IDAU, the GTZC can protect legacy memories and peripherals against non-secure transactions coming from other masters than the Cortex-M33..

Figure 6. Global TrustZone® framework and TrustZone® awareness



Securing peripherals with TZSC

When the TrustZone® security is active, a peripheral is either securable through global TrustZone® controller (GTZC) or is natively TrustZone®-aware, as shown in *Figure 6*. More specifically:

- A securable peripheral or memory is protected by an AHB/APB firewall gate that is controlled by the TrustZone® security controller (TZSC)
- A TrustZone®-aware peripheral or memory is connected directly to AHB or APB interconnect, implementing a specific TrustZone® behavior such as a subset of secure registers or a secure memory area.

When a securable peripheral is made secure-only with GTZC, if this peripheral is master on the interconnect it automatically issues secure transactions. SDMMC is an example of securable master. TrustZone®-aware AHB masters like Cortex®-M33 or DMAs drive secure signal in the AHB interconnect according to their security mode, independently to the GTZC.

Note: Like with TrustZone® a peripheral can be made privileged-only with TZSC. In this case, if this peripheral is master on the interconnect, it automatically issues privileged transactions

Securing memories with TZSC and MPCBB

The TZSC block in GTZC provides the capability to manage the security for all securable external memories (including SRAM devices), programming the MPCWM instances as defined in *Table 10*.

Table 10. MPCWMx instances

Memory	MPC instance	Type of filtering	Number of regions	Default security	on-the-fly decryption (1)
OCTOSPI	MPCWM1	Non secure region (watermarks)	2	Secure(2)	yes
FMC_NOR bank	MPCWM2		2		no
FMC_NAND bank	MPCWM3		1		no

1. Using OTDEC peripheral.
2. Assuming TrustZone® is activated on the device, non-secure otherwise.

The MPCBB instances in GTZC provide the capability to configure the security of embedded SRAM blocks, as defined in *Table 11*.

Table 11. MPCBBx instances

Memory	MPC instance	Type of filtering	Memory size (kBytes)	Block size (Bytes)	Number of blocks	Default security
SRAM1	MPCBB1	Block based, managing security and privilege	192	256(1)	768	secure(2)
SRAM2	MPCBB2		64		256	

1. Blocks are grouped in superblocks of 32 consecutive blocks, to manage configuration locking.
2. Assuming TrustZone® is activated on the device, non-secure otherwise.

Applying GTZC configurations

The TZSC and MPCBB blocks can be used in one of the following ways:

- Statically programmed during secure boot, locked and not changed afterwards.
- Dynamically re-programmed using specific application code or real-time kernel.

When dynamic option is selected and the configuration is not locked:

- MPCBB secure blocks or MPCWM non-secure regions size can be changed by secure software.
- Secure (respectively privilege) state of each peripheral can be changed writing to GTZC_TZSC_SECCFRGx (respectively GTZC_TZSC_PRIVCFGRx) registers.

Securing peripherals with TZSC

The TZSC block in GTZC provides the capability to manage the security and the privilege for all securable peripherals. The list of those peripherals can be found in *Section 5: Global TrustZone® controller (GTZC)*.

Note: When TrustZone® is deactivated, resource isolation hardware GTZC can still be used to isolate peripherals to privileged code only.

When TrustZone® is activated, peripherals are set as non-secure and non-privilege after reset.

TrustZone®-aware peripheral list

STM32L552xx and STM32L562xx devices include the following TrustZone®-aware peripherals. The way illegal accesses to those peripherals are monitored through TZIC registers is described in *Section 5: Global TrustZone® controller (GTZC)*.

- GPIOA to GPIOH
- MPCBB1 (SRAM1) and MPCBB2 (SRAM2)
- TZIC and TZSC (GTZC blocks)
- OTFDEC, writable only in secure if TZEN=1
- EXTI
- Flash memory
- RCC and PWR
- DMA1, DMA2 and DMAMUX
- SYSCFG registers
- RTC and TAMP

For more details please refer to *Section 4.6.6: Managing security in TrustZone®-aware peripherals*.

TrustZone® illegal access controller (TZIC)

The TZIC block in GTZC gathers all illegal access events originated from sources either protected by GTZC or TrustZone®-aware, generating one global secure interrupt towards the NVIC.

TZIC is available only when the system is TrustZone® enabled (TZEN = 1). All accesses to TZIC registers must be secured.

For each illegal event source, a status flag and a clear bit exist. Each illegal event can be masked, not generating an interrupt toward the NVIC.

Note: By default all events are masked.

4.6.6 Managing security in TrustZone®-aware peripherals

This section gives more details on the way security is implemented in the TrustZone®-aware peripherals listed on *Section 4.6.5*.

Embedded Flash

When the TrustZone® security is enabled through option bytes (TZEN = 1), the whole Flash memory is secure after reset and the following protections, shown on *Figure 7*, are available to the application:

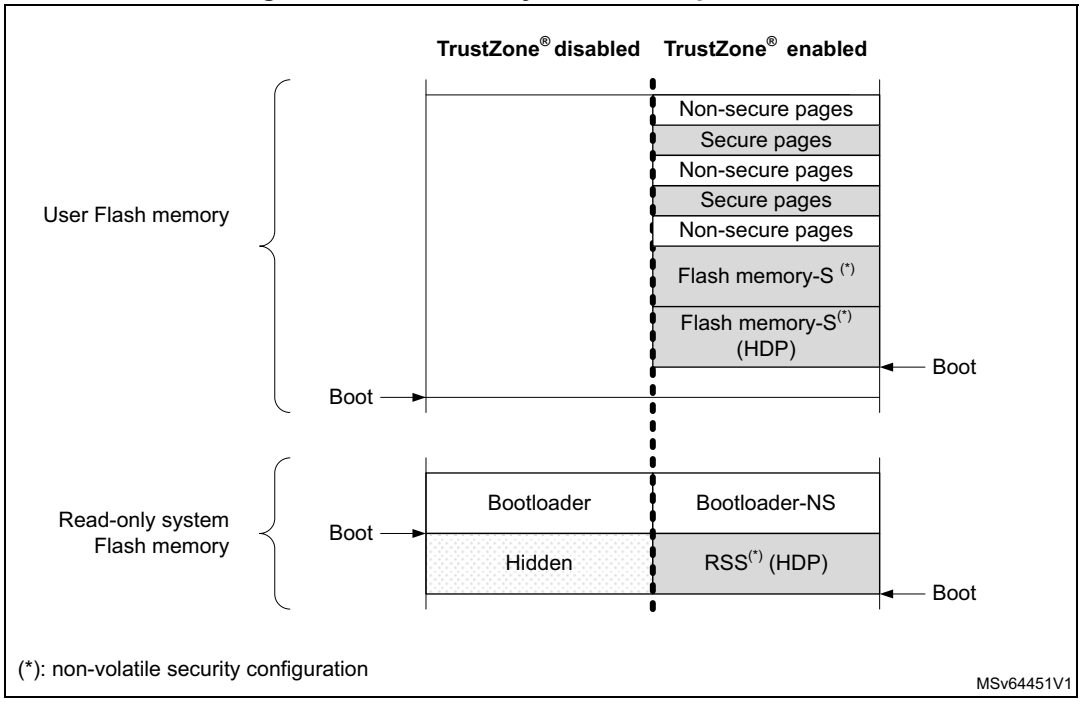
- Non-volatile user secure areas, defined with non-volatile secure user option bytes
  - Watermark-based secure only area (x2 in dual bank configuration)
  - Secure hide protection (HDP) area, stickily hidden after boot (x2 in dual bank configuration)
- Volatile user secure pages, defined with volatile secure registers (lost after reset)
  - Any page set as non-secure (example: outside watermark-based secure only area), can be set as secure on-the-fly using the block-based configuration registers

Note: All areas aligned on Flash memory page granularity.

Flash memory area can be configured as secure while they are tagged as non-secure in Cortex®-M33 IDAU/SAU. In this case non-secure accesses by the CPU to the Flash memory is denied.

Erase or program operation can be performed in secure or non-secure mode with associated configuration bits.

Figure 7. Flash memory TrustZone® protections



As shown above, when TrustZone® is activated (TZEN=1) the application code can use the HDP area that is part of the Flash watermark-based secure area. Indeed, when application sets HDPxACCDIS bit, data read, write and instruction fetch on this hide protection area are denied until next system reset. For example, the software code in the secure Flash hide protected area can be executed only once, with any further access to this area denied until next system reset. Additionally, any Flash memory page belonging to an active HDP area cannot be erased anymore.

When TrustZone® is disabled (TZEN=0) the volatile/non-volatile secure areas features are deactivated and all secure registers are RAZ/WI.

See *Section 6: Embedded Flash memory (FLASH)* for details.

On-the-fly encryption/decryption (OTFDEC)

When the TrustZone® security is activated (TZEN=1) the OTFDEC peripheral can only be initialized by secure applications. Each of the four encrypted regions, once the configuration is confirmed, can be write-locked until next power-on-reset.

Note: Any application (secure or non-secure) can verify the initialization context of each OTFDEC region (including CRC of the keys), by reading the peripheral registers.

Key registers in OTFDEC are write-only.

See *Section 4.10.3* for more details on this cryptographic engine.

DMA and DMAMUX

When a DMA channel is defined as secure (SECM = 1 in DMA_CCRx registers), the source and destination transfers can be independently set as secure or non-secure by a secure application using SSEC and DSEC bits in DMA_CCRx registers. *Table 13* summarizes these security options available in each DMA channel.

Note: Secure (resp. non-secure) DMA channel and associated DMAMUX is programmed by a secure (resp. secure or non-secure) application.

This feature is not available when TrustZone® is disabled.

Table 12. DMA channel usage (security) (1)

	Secure DMA channel (SECM = 1)		Non-secure DMA channel (SECM = 0)	
	Secure source	**Non-secure source**	**Secure source**	**Non-secure source**
Secure destination	OK	OK(2)	Transfer blocked	
Non-secure destination	OK(3)	OK(4)	Transfer blocked	OK

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return zeros. Also an illegal access event to TZIC is automatically triggered by the memory/peripheral used as source or destination.
2. If the source is a memory the transfer is only possible if SSEC = 0, otherwise the transfer is blocked.
3. If the destination is a memory the transfer is only possible if DSEC = 0, otherwise the transfer is blocked.
4. If the transfer is memory-to-memory, the transfer is only possible if SSEC = 0 and DSEC = 0, otherwise the transfer is blocked.

Similarly, when a DMA channel is defined as privileged (PRIV = 1 in DMA_CCRx register), special rules apply when accessing privileged/unprivileged source or destination. Those rules are summarized on *Table 13*.

Note: A privileged (resp. unprivileged) DMA channel and associated DMAMUX is programmed by a privileged (resp. privileged or unprivileged) application.

Table 13. DMA channel usage (privilege) (1)

	Privileged DMA channel (PRIV=1)		Non-privileged DMA channel (PRIV=0)	
	Privileged source	Non-privileged source	Privileged source	Non-privileged source
Privileged destination	OK	OK	Transfer blocked	
Non-privileged destination	OK	OK	Transfer blocked	OK

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return zeros.

Note: When a DMA transfer error occurs during a DMA read or write access, the faulty channel x is automatically disabled, and TEIFx bit is set in DMA_ISR register.

See *Section 14: Direct memory access controller (DMA)* and *Section 15: DMA request multiplexer (DMAMUX)* for details.

Power control (PWR)

When the TrustZone® security is activated (TZEN = 1), the selected PWR registers can be secured through the PWR_SECCFGR register, protecting the following PWR features:

- Low-power mode setup
- Wakeup (WKUP) pins definition
- Voltage detection and monitoring
- VBAT mode setup

Other PWR configuration bits becomes secure when:

- the system clock selection is secure in RCC. In this case the voltage scaling (VOS) configuration becomes secure.
- a GPIO is configured as secure. In this case its corresponding bit for pull-up/pull-down configuration in Standby mode becomes secure.
- the TrustZone®-aware RTC is configured as secure. In this case the backup domain write protection bit (DBP) becomes secure.
- the USB Type-C™ / USB power delivery interface (UCPD) is configured as secure in TZSC. In this case UCPD bits in PWR becomes secure.

See *Section 8: Power control (PWR)* for details.

Secure clock and reset (RCC)

When the TrustZone® security is activated (TZEN = 1) and security is enabled in the RCC, the bits controlling the peripheral clocks and resets becomes TrustZone®-aware:

- If the peripheral is securable the peripheral clock and reset bits become secure if the peripheral is programmed as secure in the TZSC.
- If the peripheral is TrustZone®-aware, the peripheral clock and reset bits become secure as soon as at least one function is configured as secured inside the peripheral.

When a peripheral is defined as secure in the RCC, the bits Enable, Reset & LPEnable become secure, and in some case the selection of clock source as well. The RCC can also secure the system clock, the system configuration, the system multiplex and the reset flag.

Note: Refer to Table 10 and Table 11 in Section 4.6.5 for the list of securable and TrustZone®-aware peripherals.

See *Section 9: Reset and clock control (RCC)* for details.

Real time clock (RTC)

Like all TrustZone®-aware peripherals, A non-secure read/write access to a secured RTC register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the RTC illegal access event is enabled in the TZIC peripheral.

After a backup domain power-on reset, all RTC registers can be read or written in both secure and non-secure modes. Secure boot code can then change this security setup, making registers Alarm A, alarm B, wakeup Timer and timestamp secure or not as needed, using RTC_SMCR register.

Note: The RTC security configuration is not affected by a system reset.

See *Section 41: Real-time clock (RTC)* for details.

Tamper and backup registers (TAMP)

Like all TrustZone®-aware peripherals, A non-secure read/write access to a secured TAMP register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the TAMP illegal access event is enabled in the TZIC peripheral.

After a backup domain power-on reset, all TAMP registers can be read or written in both secure and non-secure modes. Secure boot code can change this security setup, making some registers secure or not as needed, using TAMP_SMCR register. More specifically:

- When TAMPDPROT=0 in the TAMP_SMCR register
  - Writing the TAMP registers is possible only in secure mode. Backup registers have their own write protection (see below).
  - Reading the TAMP registers (with the exception of TAMP_SMCR, TAMP_PRIVCR and TAMP_MISR) returns zero if the access is non-secure. Backup registers have their own read protection (see below).
- Backup registers in TAMP have three protection zones configured in BKPRWDPROT[7:0] and BKPWDPROT[7:0] registers:
  - Protection zone 1 is read non-secure, write non-secure
  - Protection zone 2 is read non-secure, write secure
  - Protection zone 3 is read secure, write secure

Note: The TAMP security configuration is not affected by a system reset.

See *Section 42: Tamper and backup registers (TAMP)* for details.

General-purpose I/Os (GPIO)

When TrustZone® security is activated (TZEN = 1), each I/O pin of GPIO port can be individually configured as secure through the GPIOx_SECCFGR registers. Only secure application can write to GPIOx_SECCFGR registers. After boot, each I/O pin of GPIO is set as secure.

When an I/O pin is configured as secure, its corresponding configuration bits for alternate function (AF), mode selection (MODE) and I/O data are RAZ/WI in case of non-secure access.

When digital alternate function is used (input/output mode), in order to protect the data transiting from/to the I/O managed by a secure peripheral, the STM32L552xx and

STM32L562xx add a secure alternate function gate on the path between the peripheral and its allocated I/Os:

- If the peripheral is secure, the I/O pin must also be secure to allow input/output of data
- If the peripheral is not secure, the connection is allowed regardless of the I/O pin state.

TrustZone®-aware logic around GPIO ports used as alternate function is summarized in *Table 14*.

Table 14. Secure Alternate function between peripherals and allocated I/Os

Security configuration		Alternate function logic		Comment
Peripheral	**Allocated I/O pin**	**Input**	**Output**	
Secure	Secure	I/O data	Peripheral data	-
Non-secure				Out of reset configuration
Secure	Non-secure	Zero	Zero	-
Non-secure		I/O data	Peripheral data	

When analog function with analog switch is used, the connection to the peripherals described in *Table 15* is blocked by hardware when the peripheral is non-secure and the I/O is secure.

Table 15. Summary of the I/Os that cannot be connected to a non-secure peripheral when secure

Peripheral	Analog function(1)	Output	Input
ADCx (x= 1, 2)	ADC12_INy (y= 1 to 16)	-	X
OPAMPx (x= 1, 2)	OPAMPx_VINy (x= 1, 2 ; y= 1, 2)	-	X
COMPx (x= 1, 2)	COMPx_INy (x= 1, 2 ; y = 1, 2)	-	X

1. To find the I/O corresponding to the signal/function on the package, refer to the product datasheet.

Finally, regarding GPIO and security, *Table 16* summarizes the list of I/Os that do not have an hardware protection linked to TrustZone®. More specifically the listed signals (input and/or outputs) are not blocked when the I/O is set as secure and the associated peripheral is non secure.

For example, when secure application sets PA4 as secure to be used as LPTIM2_OUT, if the DAC peripheral is non-secure it can be programmed to *output data* to PA4, potentially causing malfunction to the secure application.

Similarly, when secure application sets PA0 as secure to be used as UART4_TX, if the TAMP peripheral is non-secure it can be programmed to capture the USART *input traffic* through the TAMP_IN signal.

Hence it is important that for each case described in *Table 16* secure application decides if a potential effect on data integrity or confidentiality is critical or not. For example, if the USART situation described above is not acceptable (data transiting on secure USART is confidential) then the secure application should configure the TAMP peripheral as secure even if it is not used by the secure application.

Note: How to make a peripheral secure is summarized on the last column.

Table 16. Summary of the I/Os that can be secured and connected to a non-secure peripheral

Peripheral	Signal (1)	Output	Input	How to set the peripheral or function as secure
DAC	DAC1_OUTx (x=1,2)	X	-	Set DAC1SEC bit in the GTZC_TZSC_SECCFGR1 register
PVD	PVD_IN	-	X	-
UCPD1	UCPD1_CCx (x=1,2)	X	X	Set UCPD1SEC bit in the GTZC_TZSC_SECCFGR1 register
	UCPD1_DBx (x=1,2)	-	X	
TSC	TSC_G1_IOy (y= 1 to 3)	-	X	Set TSCSEC bit in the GTZC_TZSC_SECCFGR2 register
	TSC_G2_IOy (y= 1 to 4)	-	X	
	TSC_G3_IOy (y= 2 to 4)	-	X	
	TSC_Gx_IOy (x=4 to 8, y=1 to 4)	-	X	
TAMP	TAMP_INx (x= 1 to 8)	-	X	Set TAMPDPROT bit in the TAMP_SMCR register
	TAMP_OUTy (x= 1 to 8)	X	-	
RTC	RTC_OUTx (x=1,2)	X	-	Set DECPROT bit in RTC_SMCR register
	RTC_REFIN	-	X	
	RTC_TS	-	X	Set TSDPROT bit in RTC_SMCR register.
PWR	WKUPx (x=1 to 5)	-	X	Set WUPxSEC bit in PWR_SECCFGR register
RCC	LSCO	X	-	Set LSESEC bit in RCC_SECCFGR register
EXTI	EXTIx (x=0 to 15)	-	X	Set SEC bit in EXTI_SECCFGR register

1. To find the I/O corresponding to the signal/function on the package, refer to the product datasheet.

For more detailed information on the topic refer to *Section 11: General-purpose I/Os (GPIO)*.

Extended interrupts and event controller (EXTI)

When the TrustZone® security is activated (TZEN = 1), the EXTI is able to protect event register bits from being modified by non-secure accesses. The protection can individually be activated per input event via the register bits in EXTI_SECCFGR registers. At EXTI level, the protection consists in preventing unauthorized write access to:

- the change of settings for the secure configurable events,
- the change of masking for the secure input events,
- the clearing of pending status for the secure input events.

The security configuration in registers EXTI_SECCFGR can be globally locked after reset in EXTI_LOCKR register.

See *Section 17: Extended interrupts and event controller (EXTI)* for details.

System configuration controller (SYSCFG)

Like all TrustZone®-aware peripherals, a non-secure read/write access to a secured SYSCFG register is RAZ/WI. Such access also generates an illegal access event that triggers a secure illegal access interrupt if the SYSCFG illegal access event is not masked in the TZIC.

See *Section 12: System configuration controller (SYSCFG)* for details.

4.6.7 Activating TrustZone® security

TrustZone® is disabled by default in all STM32L552xx and STM32L562xx devices. It can be activated by setting the TZEN option bit in FLASH_OPTR when RDP level = 0. Once TZEN has changed from 0 to 1 the default security state after reset is always the following:

- CPU subsystem
  - Cortex®-M33 exits reset in secure state, hence the boot address must point toward a secure memory area
  - All interrupt sources are secure (in NVIC)
  - The memory mapped viewed by the CPU through IDAU/SAU is fully secure
- Embedded Flash memory
  - Flash memory non-volatile secure areas are defined with non-volatile registers SECWMxR (where x = 1 or 2).
  - Volatile block-based security attributions of the Flash memory are non-secure.
- Embedded SRAM memories
  - All SRAMs are secure, as defined in GTZC/MPCBB (see *Section 4.6.5*). Secure boot code can change this security setup, making blocks secure or not as needed.
- External memories
  - All memory devices connected to FSMC and OCTOSPIs are secure, as defined in GTZC/MPCWM (see *Section 4.6.5*). Secure boot code can change this security setup, making components secure or not as needed.
- All GPIOs are secure
- All DMA channels are non-secure
- Backup registers are non-secure
- About peripherals and GTZC
  - Securable peripherals are non-secure and unprivileged
  - TrustZone®-aware peripherals are non-secure, with their secure configuration registers being secure.
  - All illegal access interrupts in GTZC/TZIC are disabled

Note: Refer to Table 10 and Table 11 in Section 4.6.5 for the list of securable and TrustZone®-aware peripherals.

4.6.8 De-activating TrustZone® security

Once TrustZone® is activated on the device it can only be deactivated during a RDP regression to level 0 (example: RDP change from 1 to 0, or from 0.5 to 0).

Note: Such RDP regression triggers the erase of embedded memories (SRAM2, flash), and the reset of all peripherals, including the on-the-fly decryption and all the crypto engines.

After the TrustZone® deactivation, most features mentioned in *Section 4.6* are no more available. More specifically:

- Non-volatile secure area of embedded Flash memory is deactivated, including the HDP area
- NVIC only manages non-secure interrupts
- All secure registers in TrustZone®-aware peripherals are RAZ/WI.

Note: When TrustZone® is disabled GTZC can still be used to configure the privilege access to securable peripherals.

For more information please refer to application AN5347 available at *www.st.com*.

4.7 Other resource isolations

These are hardware mechanisms offering an additional level of isolation on top of TrustZone® technology.

4.7.1 Temporal isolation using secure hide protection (HDP)

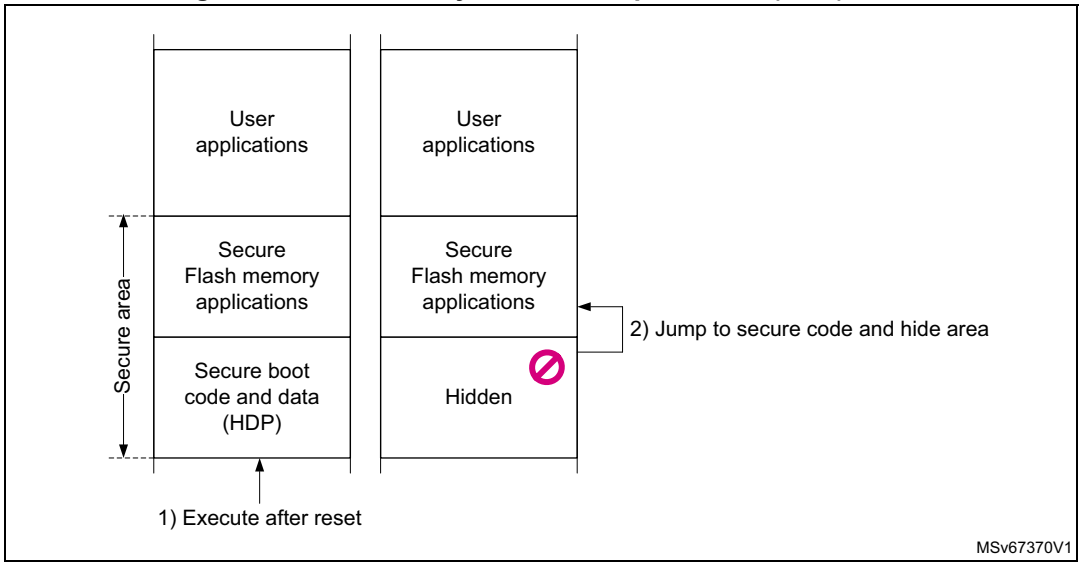
The STM32L552xx and STM32L562xx embedded Flash memory allows to define one hide protection (HDP) area per watermarked-secure area of each bank. The code executed in HDP area, with its related data and keys, can be hidden after boot until the next system reset. Hide protection principle is pictured on *Figure 8*.

The number of HDP area, and its granularity, depends on the DBANK mode:

- in single-bank mode (DBANK=0) two HDP area can be defined, with the granularity of a 4 Kbytes page.
- in dual-bank mode (DBANK=1) one HDP area per bank can be defined, with the granularity of a 2 Kbytes page.

When HDPxEN and HDPxACCDIS bits are set, data read, write and instruction fetch on the area defined by SECWMx_STRT and HDPx_PEND are denied until next device reset.

Figure 8. Flash memory secure hide protection (HDP) area



Note: Bank erase aborts when it contains a write-protected area (WRP or HDP area).

HDP area can be resized by secure application if the area is not hidden and if RDP level is different than 2.

4.8 Secure execution

4.8.1 Introduction

Through a mix of special software and hardware features, STM32L552xx and STM32L562xx devices ensure the correct operation of their functions against abnormal situations caused by programmer errors, software attacks through network access or local attempt for tampering code execution.

This section describes the hardware features specifically designed for secure execution.

4.8.2 Memory protection unit (MPU)

The Cortex®-M in STM32L552xx and STM32L562xx devices includes a memory protection unit (MPU) that can restrict the read and write accesses to memory regions (including regions mapped to peripherals), based on one or more of the following parameters

- Cortex®-M operating mode (privileged, unprivileged)
- Data/instruction fetch

The memory map and the programming of the non-secure and secure MPUs split memory into regions (up to eight per MPU). Secure MPU is only available when TrustZone is activated.

4.8.3 Embedded Flash memory write protection

The embedded Flash memory write protection (WRP) prevents illegals or unwanted write/erase to special sections of the embedded Flash memory user area (system area is permanently write protected).

Write protected area is defined through the option bytes, writing the start and end addresses. More specifically, in STM32L552xx and STM32L562xx:

- In single-bank mode (DBANK = 0) four write-protected areas can be defined, with the granularity of a 4 Kbytes page.
- In dual-bank mode (DBANK = 1) two write-protected areas can be defined in each bank, with the granularity of a 2 Kbytes page.

WRP areas can be modified through option byte changes while RDP level is less than 2.

Note: Bank erase aborts when it contains a write-protected area (WRP or HDP area)

4.8.4 Tamper detection and response

Principle

STM32L552xx and STM32L562xx devices include active protection of critical security assets against temperature, voltage and frequency attacks. More specifically, it features:

- erasure of device secrets upon tamper detection
- improved guarantee of safe execution for the CPU and its associated security peripherals, including:
  - out-of-range voltage (example: V_{BAT}, V_{DDA}), temperature and clocking (LSE) detection
  - security watchdog IWDG clocked by the internal oscillator LSI
  - possible selection of internal oscillator HSI as system clock
- power supply protection
  - RTC/TAMP domain powered automatically with VDD or VBAT

See *Section 42: Tamper and backup registers (TAMP)* for details.

Tamper detection sources

The device features three active tamper inputs (TAMP_IN1,2,3) associated to one active output tamper pin (TAMP_OUT2), available in all power modes (including VBAT mode).

The device also has a number of internal tamper sources, as described in *Table 17*.

Note: Timestamps are automatically generated when a tamper event occurs.

Table 17. Internal tampers in TAMP

Tamper input	NOER bit	Tamper source
itamp1	TAMP_CR3[0]	V_{DD} upper voltage threshold monitoring
itamp2	TAMP_CR3[1]	Temperature monitoring
itamp3	TAMP_CR3[2]	LSE monitoring
itamp4	-	*not used*
itamp5	TAMP_CR3[4]	RTC calendar overflow (rtc_calovf)

Table 17. Internal tampers in TAMP (continued)

Tamper input	NOER bit	Tamper source
itamp6	-	*not used*
itamp7	-	*not applicable*
itamp8	TAMP_CR3[7]	Monotonic counter overflow (generated internally)

Response to tampers

Each source of tamper in the device can be configured to trigger the following events:

- Generate an interrupt, capable of waking up the device from Stop and Standby modes (see TAMPxMSK bits in TAMP_CR2 register)
- Generate a hardware trigger for the low-power timers
- Erase of device secrets if corresponding TAMPxNOER bit is cleared in TAMP_CR2 register (for tamper pins) or TAMP_CR3 register (for internal tamper). These erasable secrets are:
  - Symmetric keys stored in backup registers (x32), in AES, HASH and in OTFDEC (encrypted Flash memory regions are read as zero)
  - Asymmetric keys stored in PKA SRAM
  - Other secrets stored in SRAM2 and CPU instruction cache memory

Note: Device secrets erase are also triggered by setting the BKERASE bit in the TAMP_CR2 register, or by performing an RDP regression as defined in Section 4.11.2: Lifecycle management with readout protection (RDP).

Device secrets are not reset by system reset or when the device wakes up from Standby mode.

Tamper detection and low power modes

The effect of low power modes on tamper detection are summarized on *Table 18*.

Table 18. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect on tamper detection features. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on tamper detection features, except for level detection with filtering and active tamper modes 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 tamper detection features, except for level detection with filtering and active tamper modes 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 tamper detection features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE. Tamper events cause the device to exit the Shutdown mode.

4.9 Secure storage

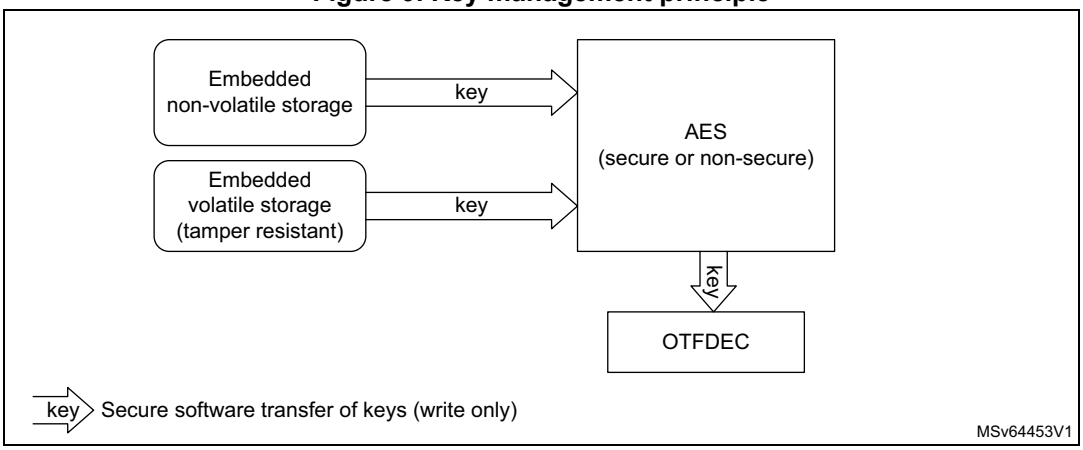
4.9.1 Introduction

A critical feature of any security system is how root keys are stored, protected, and provisioned. Such keys are typically used for loading a boot image, or handling of critical user data.

Figure 9 shows how key management service application can use the AES engine for example to compute external image decryption keys. Embedded non-volatile key can be stored in the secure HDP area (see *Section 4.7.1*), while volatile key storage consists in the battery-powered, tamper-protected SRAM or registers in TrustZone®-aware TAMP peripheral.

Details on tamper protection is found in *Section 4.8.4*, while TrustZone® features of TAMP is briefly described in *Section 4.6.6*.

Figure 9. Key management principle



4.9.2 Unique ID

The STM32L552xx and STM32L562xx store a 96-bit ID that is unique to each device. It is stored at the address **0x0BFA 0590**.

Application services can use this unique identity key to identify the product in the cloud network, or make it difficult for counterfeit devices or clones to inject untrusted data into the network.

4.10 Crypto engines

4.10.1 Introduction

STM32L552xx and STM32L562xx devices implement state-of-the-art cryptographic algorithms featuring key sizes and computing protection as recommended by national security agencies such as NIST for the U.S.A, BSI for Germany or ANSSI for France. Those algorithms are used to support privacy, authentication, integrity, entropy and identity attestation.

The crypto engines embedded in STM32 reduces weaknesses on the implementation of critical cryptographic functions, preventing for example the use of weak cryptographic algorithms and key sizes. They also enable lower processing times and lower power consumption when performing cryptographic operations, offloading those computations from Cortex®-M33. This is especially true for asymmetric cryptography.

For product certification purpose, ST can provides certified device information on how these security functions are implemented and validated.

For more information on crypto engine processing times, refer to their respective sections in the reference manual.

4.10.2 Crypto engines features

Table 19 lists the accelerated cryptographic operations available in the STM32L552xx and STM32L562xx devices.

Note: Additional operations can be added using firmware.

Public key accelerator can accelerate asymmetric crypto operations (like key pair generation, ECC scalar multiplication, point on curve check). See Section 32: Public key accelerator (PKA) for details.

Table 19. Accelerated cryptographic operations

Operations	Algo-rithm	Specification	Key lengths (in bit)	Modes
Get entropy	RNG	NIST SP800-90B(1)	N/A	N/A
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D	128, 256	GCM, CCM
Cipher-based message authentication code		NIST SP800-38D	128, 256	GMAC
Checksum	MD5	IETF RFC 1321	n/a	Digest 128-bit
	SHA-1	FIPS PUB 180-4		Digest 160-bit
Cryptographic hash	SHA-2			SHA-224, SHA-256
Keyed-hashing for message authentication	HMAC	FIPS PUB 198-1 IETF RFC 2104	short, long (>64 bytes)	-
Encryption/decryption key-pair generation	RSA	IETF RFC 8017 NIST SP800-56B	up to 3136	RSAES-OAEP

Table 19. Accelerated cryptographic operations (continued)

Operations	Algo-rithm	Specification	Key lengths (in bit)	Modes
Signature with hashing Signature verification	RSA	IETF RFC 8017 FIPS PUB 186-4	up to 3136	PKCS1-v1_5, PSS
	ECDSA	ANSI X9.62 IETF RFC 7027 FIPS PUB 186-4	up to 640	Nist: P-256, P-384, P-521,... Brainpool: P256r1/t1, P384r1/t1, P512r1/t1,... SEC2: secp256k1 OSCCA SM2
Key agreement	ECDH	ANSI X9.42		

1. Certifiable using STMicroelectronics reviewed documents.

4.10.3 On-the-fly decryption engine (OTFDEC)

OTFDEC TrustZone®-aware peripheral proposes on-the-fly decryption of encrypted images stored on external Flash memory, connected through OCTOSPI peripheral. This decryption process introduces almost no additional cycle overhead when standard NOR Flash memory is used. OTFDEC can also be used to encrypt Flash memory images on the device, for example to encrypt with a device unique secret key.

When a tamper event is confirmed in TAMP peripheral, all OTFDEC keys are erased and encrypted regions read as zero until OTFDEC is properly initialized again.

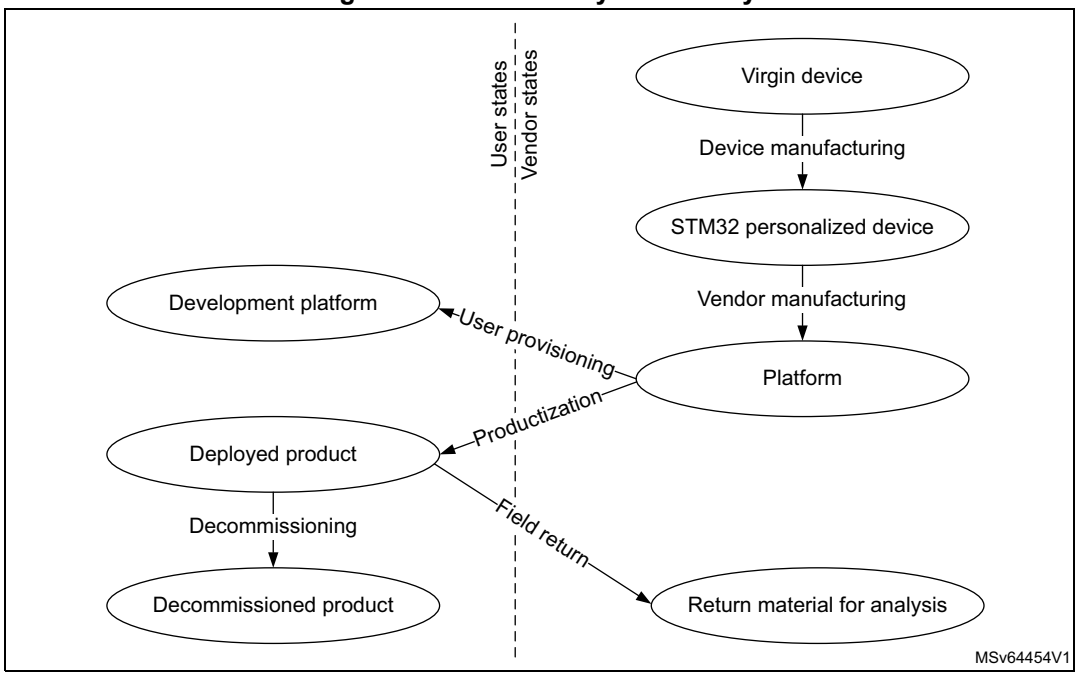
Typical usage of OTFDEC can be found in *Section 4.13.3: Software intellectual property protection with OTFDEC*. For more details on the programming of the peripheral please refer to *Section 31: On-the-fly decryption engine (OTFDEC)*.

4.11 Product Lifecycle

4.11.1 Introduction

A typical IoT device lifecycle is summarized in *Figure 10*. For each step, STM32L552xx and STM32L562xx devices propose secure lifecycle management mechanisms embedded in the hardware.

Figure 10. Device lifecycle security



More details on the various phases and associated transitions, found either at the vendor or end user premises, are summarized on *Table 20*.

Table 20. Main product lifecycle transitions

Transitions	Description
Device manufacturing	STMicroelectronics creates new STM32 devices, always checking for manufacturing defects. During this process STM32 is provisioned with ROM firmware, secure firmware install (SFI) unique key pair, and a public ID.
Vendor manufacturing	One (or more) vendor is responsible for the platform assembly, initialization, and provisioning before delivery to the end user. This end user can use the final product (“productization” transition) or he/she can use the platform for software development (“user provisioning” transition).
Productization	The end user gets a product ready for use. All security functions of the platform are enabled, the debugging/testing features are restricted/disabled, and unique boot entry to immutable code is enforced.
User provisioning	Platform vendor prepares an individual platform for development, not to be connected to a production cloud network.
Field return or decommissioning	Those are one way transitions, with devices kept in user premises or returned to the manufacturer. In both cases all data including user data are destroyed, therefore the device loses the ability to operate securely (e.g. connecting to a managed IoT network).

The features described hereafter contribute to securing the device lifecycle.

4.11.2 Lifecycle management with readout protection (RDP)

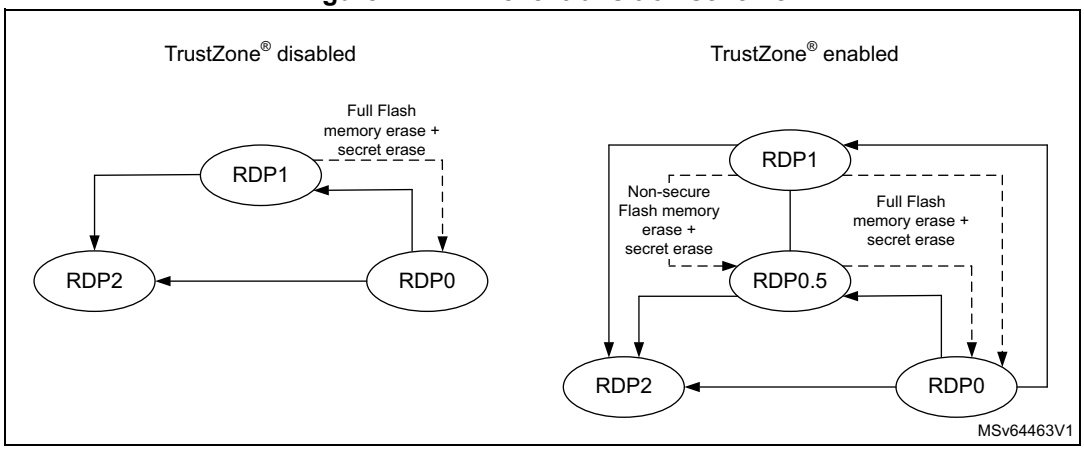
The readout protection mechanism (full hardware feature) controls the access to the STM32L552xx and STM32L562xx debug, test and provisioned secrets, as summarized on *Table 21*. The supported transitions, summarized on *Figure 11*, can be requested (when available) through the debug interface or via the system bootloader.

Table 21. Typical product lifecycle phases

RDP protection level		Debug	Comments
Level 0	Device is open	Secure(1) and non-secure	Boot address must target a secure area when TrustZone® is enabled (secure SRAM, secure Flash memory, RSS in system Flash memory).
Level 0.5(2)	Device is partially closed (closed-secure)	Non-secure only	Boot address must target a secure area when TrustZone® is enabled (secure user or system Flash memory). boot on SRAM is not permitted. Access to non-secure Flash memory is allowed when debug is connected.
Level 1	Device memories are protected	Non-secure only (conditioned)	Boot address must target the secure user Flash memory. Accesses to non-secure Flash memory, encrypted Flash memory(3), SRAM2 and backup registers are not allowed when debug is connected.
Level 2	Device is closed	None (JTAG fuse)	Boot address must target the user Flash memory (secure if TZEN=1). Option bytes are read-only, hence RDP level 2 cannot be changed.

1. Debug is not available when executing RSS code.
2. Only applicable when TrustZone® security is activated in the product.
3. External Flash memory area decrypted on-the-fly with OTFDEC peripheral.

Figure 11. RDP level transition scheme



As shown on *Figure 11*, the user Flash memory is automatically erased, either partially or in totality, during a RDP regression. During the transition from RDP1 to RDP0.5 only non-secure embedded Flash memory is erased, keeping functional for example the secure boot and the secure firmware update. In all regressions OTP area in Flash memory is kept, and device secrets are erased, hence no secrets shall be stored in OTP as they are revealed after a regression to RDP0. Those secrets, erased as response to tamper, are defined in *Section 4.8.4: Tamper detection and response*.

Note: Enabling TrustZone® using option byte TZEN is only possible when RDP level is 0.

For more details on RDP please refer to *Section 6: Embedded Flash memory (FLASH)*.

4.11.3 Recommended option byte settings

Most of the time, the user threat model focuses mainly on software attacks, in this case, it may be sufficient to keep the RDP level 1 as device protection.

For a more aggressive threat model, where user may fear physical attacks on the STM32 device, it is recommended to optimize the level of security by setting the RDP level 2. The recommended settings are detailed below:

If TrustZone® is disabled (TZEN=0) it is recommended to set the following option bytes:

- RDP = level 2
- Non-secure boot address option bytes set in user Flash memory

If TrustZone® is enabled (TZEN=1) it is recommended to set the following option bytes:

- RDP = level 2
- Boot_lock = 1
- Secure boot address option bytes set in user Flash memory

4.12 Access controlled debug

4.12.1 Introduction

The device restricts access to embedded debug features, in order to guarantee the confidentiality of customer assets against unauthorized usage of debug & trace features.

4.12.2 Debug protection with readout protection (RDP)

As described in *Section 4.11.2* the hardware readout protection (RDP) mechanism automatically controls the accesses to the device debug and tests. The protection of these debug features are defined in *Table 22*

Table 22. Debug protection with RDP

RDP protection level		Debug features protection
Level 0	Device is open	Any debug(1)
Level 0.5(2)	Device is partially closed	Secure debug is no more available
Level 1	Device memories are protected	Non-secure debug can no longer debug code & data stored in embedded Flash memory, encrypted external Flash memory(3), SRAM2 and backup registers
Level 2	Device is closed	JTAG is physically deactivated

1. Including ST engineering test modes, used for field returns.
2. Only applicable when TrustZone® security is activated in the product.
3. External Flash memory area decrypted on-the-fly with OTFDEC peripheral.

4.13 Software intellectual property protection and collaborative development

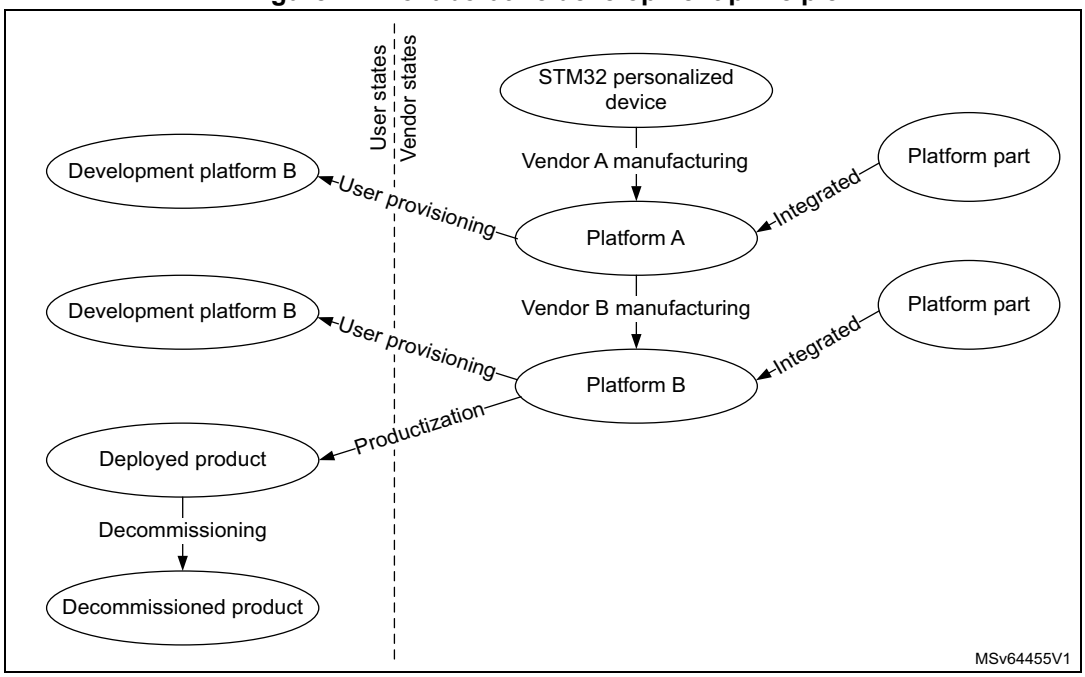
4.13.1 Introduction

Thanks to software intellectual property protection and collaborative model the STM32L552xx and STM32L562xx devices allow to integrate and implement with third-party libraries innovative solutions.

Collaborative development is summarized on *Figure 12*. Starting from a personalized device sold by STMicroelectronics, a vendor A can integrate a portion of hardware and software on a platform A, that can then be used by a vendor B that will do the same before deploying a final product to the end users.

Note: Each platform vendor can provision individual platforms for development, not to be connected to a production cloud network ("Development Platform X").

Figure 12. Collaborative development principle



The features described hereafter contribute to securing the software intellectual property within such a collaborative development.

4.13.2 Software intellectual property protection with readout protection (RDP)

As described in *Section 4.11.2* the hardware readout protection (RDP) mechanism automatically controls the accesses to secrets provisioned in the device. The protection of these secrets are defined in *Table 23*.

Table 23. Software intellectual property protection with RDP

RDP protection level		Secrets protection
Level 0	Device is open	No special protections.
Level 0.5(1)	Device is partially closed	All peripherals and memories mapped as secure during secure boot cannot be dumped, debugged or traced
Level 1	Device memories are protected	Data and code stored in embedded Flash memory, encrypted external Flash memory(2), SRAM2 and backup registers are no more accessible via debugger.
Level 2	Device is closed	All data and code stored in the device or encrypted in external Flash memory cannot be dumped clear-text, debugged or traced.

1. Only applicable when TrustZone® security is activated in the product.
2. External Flash memory area decrypted on-the-fly with OTFDEC peripheral.

4.13.3 Software intellectual property protection with OTFDEC

As described in *Section 4.10.3* the OTFDEC peripheral associated with the OCTOSPI is able to decrypt on-the-fly encrypted images stored in external SPI Flash memory devices.

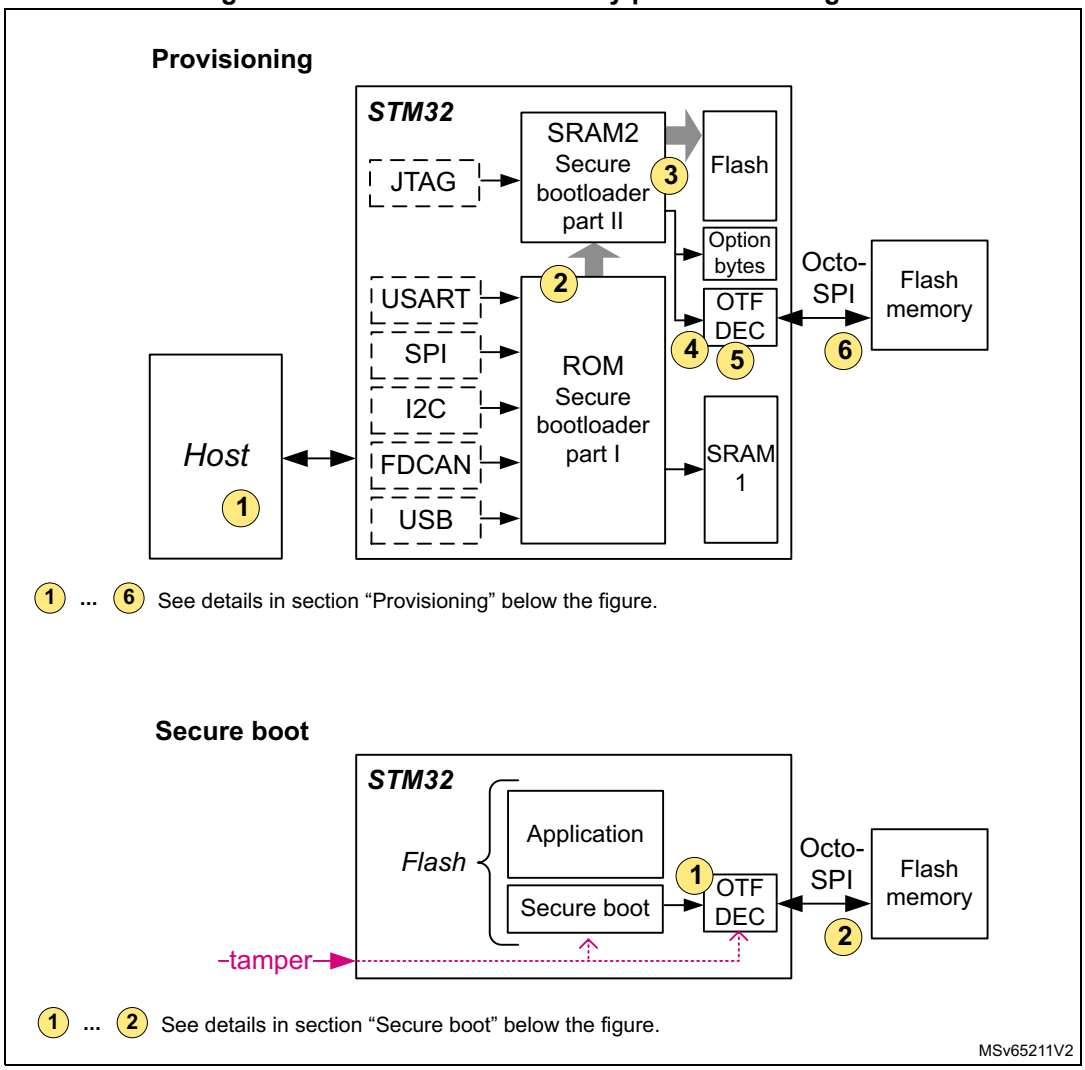
Thanks to this feature STM32L562xx devices allow the installation of intellectual properties:

- over the air, with the image already encrypted with a key provisioned in the device, or
- through a provisioning host located in a trusted or an non-trusted environment/facility.

Figure 13 illustrates this last case with the provisioning, in a non-trusted environment, of software intellectual properties both in embedded Flash memory and in external SPI Flash memory (encrypted).

Note: Since OTFDEC is using AES in counter mode (CTR) to achieve the lowest possible latency, each time the content of one encrypted region is changed the corresponding cryptographic context (key or initialization vector) must be changed. This constraint makes OTFDEC suitable to decrypt read-only data or code, stored in external NOR Flash memory.

Figure 13. External Flash memory protection using SFI



Provisioning

Assuming the device is virgin, the first step is to provision both Flash memories, as detailed below:

1. User creates an SFI image, composed of:
   - Encrypted internal firmware and data (including external Flash memory drivers)
   - Encrypted external firmware and data AES key (up to 4)
   - Encrypted external firmware and data image
2. The secure bootloader stored in system memory loads the second part of the secure-bootloader in SRAM2 through the supported communication ports (USART, SPI, I2C, FDCAN, USB and JTAG). This second part runs in secure SRAM2 and is responsible

for executing the SFI process, applying the SFI protocol thanks to the commands received through the above mentioned supported communication ports.

3. Internal Flash memory is programmed with decrypted option bytes, internal firmware and data, and external firmware and data AES key(s). Alternatively, device unique external firmware AES keys could be used instead of such global keys.
4. OTFDEC is properly initialized with encrypted region(s) information, including the corresponding external firmware and data AES key.
5. Running the SFI process, chunks of encrypted external firmware and data image are decrypted in the device, then re-encrypted in OTFDEC.
6. After chunk OTFDEC re-encryption, user external Flash memory programmer is responsible for programming those last encrypted chunks to external SPI Flash memories through OCTOSPI peripheral.

Secure boot

After provisioning, each time the device initializes on trusted firmware, the following actions are required:

1. Secure boot firmware executes, programming the external firmware and data AES key(s) to OTFDEC write-only key registers, along with the other needed information.
2. Application reads or executes the encrypted external firmware and data through OCTOSPI memory mapped mode, unless a tamper event is detected. In this case all OTFDEC keys are erased and encrypted regions read as zero until OTFDEC is properly initialized again.

For more information on above secure firmware install (SFI) solutions for STM32L552xx and STM32L562xx devices please refer to AN4992 on STMicroelectronics website.

4.13.4 Other software intellectual property protections

STM32L552xx and STM32L562xx devices additional protections to software intellectual property is:

- Invasive attacks such as physical tampering or perturbation are countered by detection then decommissioning of the device before the detected attack succeeds.

5 Global TrustZone® controller (GTZC)

5.1 GTZC introduction

This section includes the description of the three following sub-blocks:

- **TZSC:** TrustZone® security controller

  This sub-block defines the secure/privilege state of slave/master peripherals. It also controls the non-secure area size for the watermark memory peripheral controller (MPCWM). The TZSC block informs some peripherals (such as RCC or GPIOs) about the secure status of each securable peripheral, by sharing with RCC and I/O logic.
- **MPCBB**: block-based memory protection controller

  This sub-block controls secure states of all blocks (256-byte pages) of the associated SRAM.
- **TZIC**: TrustZone illegal access controller

  This sub-block gathers all illegal access events in the system and generates a secure interrupt towards NVIC.

These sub-blocks are used to configure TrustZone system security in a product having bus agents with programmable-security and privileged attributes (securable) such as:

- on-chip RAM with programmable secure blocks (pages)
- AHB and APB peripherals with programmable security and/or privilege access
- AHB master granted as secure and/or privilege
- off-chip memories with secure areas

5.2 GTZC main features

GTZC main features are listed below:

- 3 independent 32-bit AHB interface for TZSC, MPCBB and TZIC
- MPCBB and TZIC accessible only with secure transactions
- Secure and non-secure access supported for priv/non-priv part of TZSC
- Set of registers to define product security settings:
  - Secure blocks for internal SRAM
  - Non-secure regions for external memories
  - Secure/privilege access mode for securable and TrustZone-aware peripherals
  - Secure/privilege access mode for securable masters

5.2.1 GTZC TrustZone system architecture

The Armv8-M supports security per TrustZone-M model with isolation between:

- a secure world, where usually security sensitive applications are run and critical resources are located; secure transactions are signaled with HNONSEC[1] = 0 on AHB bus
- a non-secure or public world (such as usual non secure and user space) where non-secure transactions are signaled with HNONSEC = 1 on AHB bus

The TrustZone architecture is extended beyond AHB and Armv8-M with:

- AHB/APB bridge used as secure gate to block or propagate secure/non-secure and privilege/non-privilege transaction towards APB agents
- peripheral protection controller (PPC) used as secure gate to block or propagate secure/non-secure and privilege/non-privilege transaction towards AHB agents
- TrustZone block-based MPC firewalls used as secure gate to filter secure/non secure access towards internal SRAMs
- Trustzone watermark MPC firewalls used as secure gate to filter secure/non secure access towards external memories

AHB and APB peripherals can be categorized as:

- **privilege**: peripherals protected by AHB/APB firewall stub that is controlled from TZSC to define privilege properties
- **secure:** peripherals always protected by an AHB/APB firewall stub. These peripherals are always secure (TZIC).
- **securable:** peripherals protected by an AHB/APB firewall stub that is controlled from TZSC to define security properties (optional)
- **non-secure and non-privilege:** peripherals connected directly to AHB/APB interconnect without any secure gate
- **TrustZone-aware**: peripherals connected directly to AHB or APB bus and implementing a specific TrustZone behavior (such as a subset of registers being secure). TrustZone-aware AHB masters always drive HNONSEC signal according to their security mode (such as Armv8-M core or DMA).

AHB securable masters can be configured in the TZSC to be secure/non-secure and/or privilege/non-privilege.

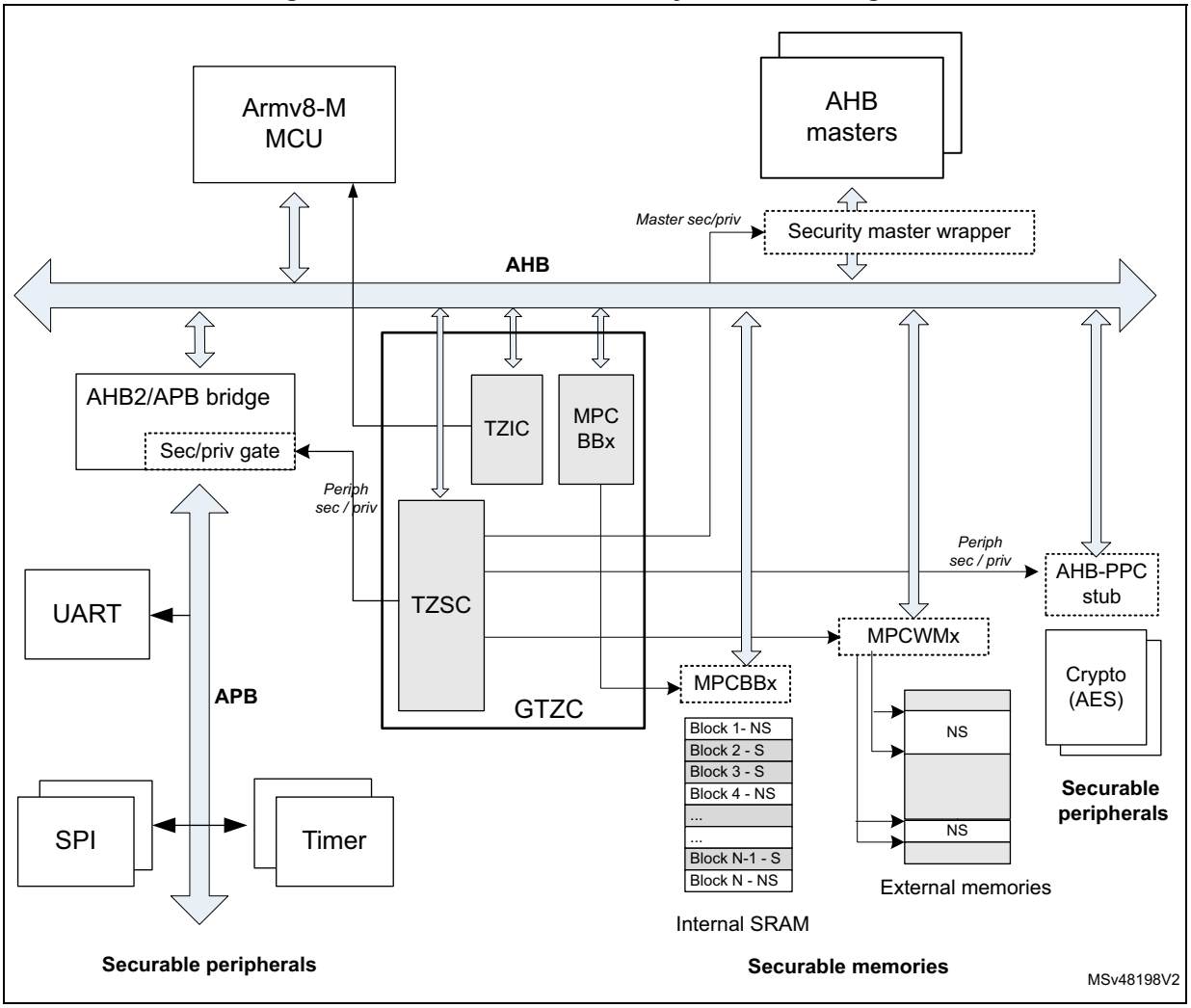
Application information

The TZSC, MPCBB and TZIC sub-blocks can be used in one of the following ways:

- programmed during secure Boot only, locked and not changed afterwards
- dynamically re-programmed when using specific application code or secure kernel (microvisor). When not locked, MPC secure blocks or region size can be changed by secure software executing from the secure FLASH memory region or secure SRAM. Same remark applies to the GTZC_TZSC_SECCFRGx and GTZC_TZSC_PRIVCFGRx registers that define secure/privilege state of each peripheral.

The Armv8-M security architecture with secure, securable and TrustZone-aware peripherals is shown in *Figure 14*.

Figure 14. GTZC in Armv8-M subsystem block diagram



5.3 GTZC functional description

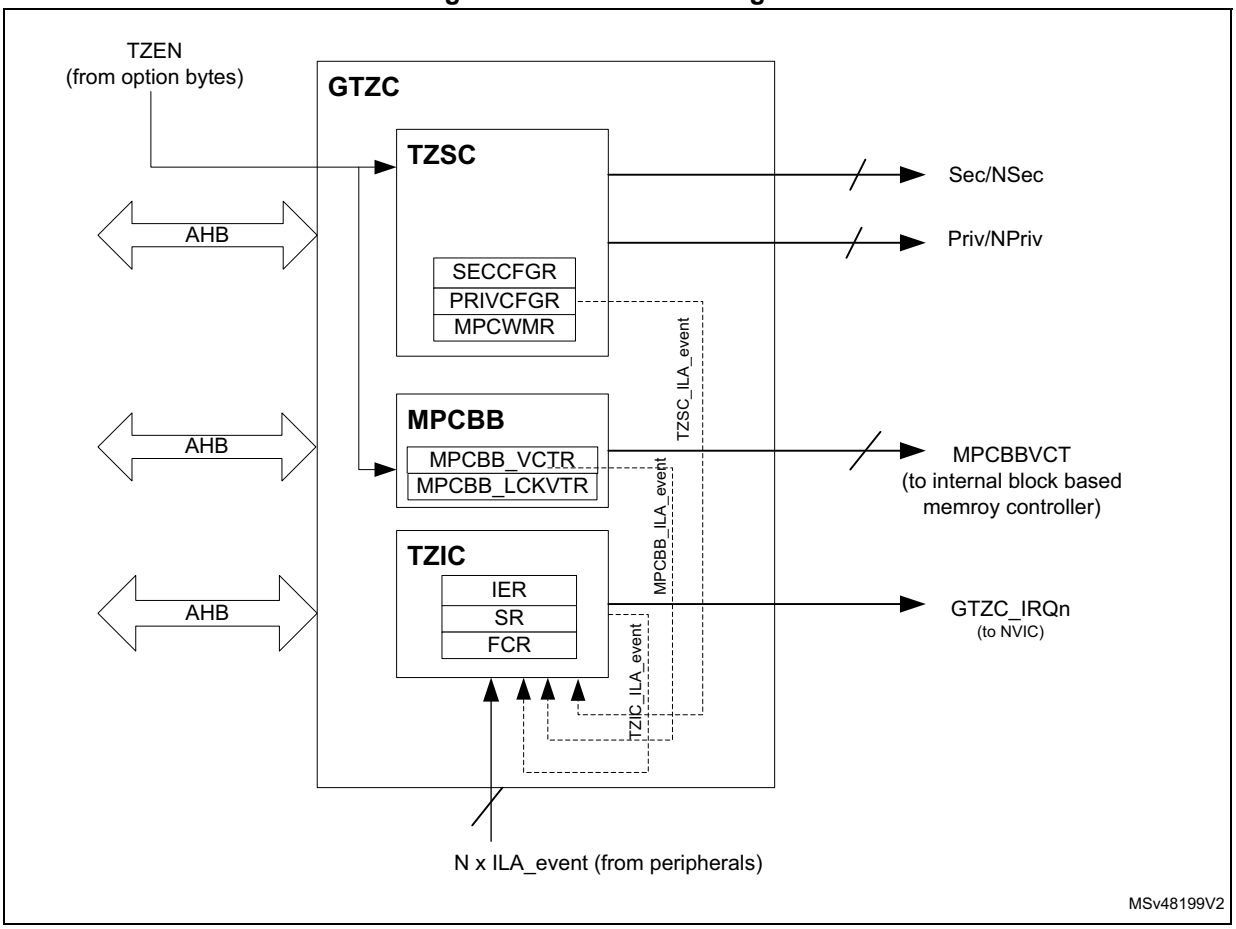
5.3.1 GTZC block diagram

Figure 15 describes the combined feature of TZSC, MPCBB and TZIC. Each sub-block is controlled by its own AHB configuration port. The TZSC defines which peripheral is secured and/or privileged.

The privilege configuration bit of a given peripheral can be modified by a secure-privilege transaction when the peripheral has been configured as secure, otherwise a privileged transaction (non-secure) is sufficient. The definition of these privilege attributes is possible even when TZEN = 0.

The secure configuration bit a given peripheral can be modified **only** with a secure-privilege transaction if the peripheral has been configured as privilege, otherwise a secure transaction (non-privileged) is sufficient.

Figure 15. GTZC block diagram



5.3.2 Illegal access definition

Three different types of illegal access exist:

- Illegal non-secure access

  Any non-secure transaction trying to write a secure resource is considered as illegal and thus the addressed resource generates an illegal access event for illegal write access, and a bus error for illegal fetch access. However some exceptions exist on transactions to secure and privilege configuration registers; these later ones authorized non-secure read access to secure registers (see GTZC_TZSC_SECCFGRx and GTZC_TZSC_PRIVCFGRx registers).

- Illegal secure access

  Any secure transaction trying to access a non-secure block in internal block-based SRAM or watermarked external memory, is considered as illegal.

  Correct settings of the TZIC allows the capture of the associated event and then generates the GTZC_IRQn interrupt to the NVIC. This applies for read, write and execute access.

Concerning the MPCBB controller, there is an option to ignore secure data read/write access on non-secure SRAM blocks, by setting the SRWILADIS configuration bit in the GTZC_MPCBBx_CR register. Secure read and write data transactions are then allowed on non-secure SRAM blocks, while secure execution access remains not allowed.

Any secure execute transaction trying to access a non-secure peripheral register or memory is considered as illegal and generates a bus error.

- Illegal non-privilege access

  Any non-privilege transaction trying to acces a privilege resource is considered as illegal. There is no illegal access event generated for this type of illegal access. The addressed resource follows a silent-fail behavior, returning all zero data for read and ignoring any write. No bus error is generated.

5.3.3 TrustZone security controller (TZSC)

This block is composed of a configurable set of registers, providing the following features:

- Control of secure and privilege state for all peripherals, done through:
  - GTZC_TZSC_SECCFGRx registers to control AHB/APB firewall stubs for the securable peripherals and/or master
  - GTZC_TZSC_PRIVCFGRx registers to control AHB/APB firewall stubs for the privileged peripherals and/or master
- For watermark memory protection controller (external memories), two independent regions can be defined and the following fields are used to program:
  - start of the first non-secure area on external memory = NSWMxSTRT1[10:0]
  - length of the first non-secure area on external memory = NSWMxLGTH1[11:0]
  - start of the second non-secure area on external memory = NSWMxSTRT2[10:0]
  - length of the second non-secure area on external memory = NSWMxLGTH2[11:0]

Note: x represents the target external memory interface (such as FMC or OCTOSPI). The total area considered as non-secure is the sum of the two independent ones. An overlap of one section over the other one has no specific effect.

Table 24 describes the characteristics of the available MPCWMx.

Table 24. MPCWMx

MPC	Type	Number of regions	Target memory interface
MPCWM1	Non secure watermark (region)	2	OCTOSPI
MPCWM2	Non secure watermark (region)	2	FMC_NOR bank
MPCWM3	Non secure watermark (region)	1	FMC_NAND bank

5.3.4 Memory protection controller - block based (MPCBB)

For block-based memory protection controller (internal SRAM), the below registers are available:

- GTZC_MPCBBx_VCTRy to program the vector defining 32 consecutive 256-byte block states, secure or non-secure (means y registers of 32 bits)
- GTZC_MPCBBx_LCKVTRy to program the lock of super block (32 blocks)

Note: *y* represents the number of super-blocks (super-block = *32 blocks of SRAM_BB_size). On the STM32L552xx and STM32L562xx devices, the SRAM size = 192 Kbytes and the block size = 256 bytes, then super-block size = 32 * 256 = 8 Kbytes and y = 192 / 8 = 24. It means **24** vector registers (32-bit) are needed to control the secure state of all the 256-byte blocks. Concerning the lock bit, only **one** 32-bit lock register is needed since a single lock bit applies on a super-block (32 * 256 bytes).*

Table 25 describes the characteristics of the available MPCBBx.

Table 25. MPCBBx

MPC	Type	Number of blocks	Target memory interface
MPCBB1	Block based (block size = 256 bytes)	768	SRAM1
MPCBB2	Block based (block size = 256 bytes)	256	SRAM2

5.3.5 TrustZone illegal access controller (TZIC)

This block concentrates all illegal access source events. It is used only when the system is TrustZone enabled (TZEN = 1).

TZIC allows the trace of which event trigged the NVIC GTZC_IRQn. Register masks (GTZC_TZIC_IERx) are available to filter unwanted event. On unmasked illegal event, the TZIC generates the GTZC_IRQn signal to NVIC that corresponds to the GTZC_IRQn secure interrupt.

For each illegal event source, a status flag and a clear bit exist (respectively within GTZC_TZIC_SRx and GTZC_TZIC_FCRx registers). The reset value of mask registers (GTZC_TZIC_IERx) is such that all events are masked.

5.3.6 Power-on/reset state

The power-on and reset state of the TZSC clear all bits of GTZC_TZSC_SECCFGRx and GTZC_TZSC_PRIVCFGRx registers to 0, which respectively means non-secure and non-privilege. Concerning the internal SRAM, the reset values of the GTZC_MPCBBx_VCTRy registers are set to 0xFFFF FFFF, making all blocks of the block-based SRAM memory secure. This is valid only when TrustZone security is enabled at system level (TZEN = 1). In the other case (legacy mode, TZEN = 0), the reset values of the GTZC_MPCBBx_VCTRy registers are set to 0x0000 0000 so that all block-based SRAM memories are non-secure.

Same thing applies for external memories, all MPCWMx_NSWMyR registers must be set to 0xFFFF FFFF, making the whole external memory non-secure. Secure Boot code can then program the security settings, making components secure or not as needed.

5.3.7 DMA requests

TZSC does **not** support any DMA interface.

5.4 GTZC events

MPCBB and TZIC are secure peripherals, thus they both systematically generate an illegal access event when accessed by a non-secure access. The TZSC is a TrustZone-aware peripheral, meaning that secure and non-secure registers co-exist within the peripheral. An

exception exists for the GTZC_TZSC_SECCFGR and GTZC_TZSC_PRIVCGFR: any read access, secure or not, are supported.

5.5 GTZC_TZSC registers

All registers are accessed only by words (32-bit).

5.5.1 GTZC_TZSC control register (GTZC_TZSC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Write-secure access only.

Read accesses are authorized for any type of transactions, secure or not, privilege or not.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LCK**: lock the configuration of TZSC items until next reset
This bit is unset by default and once set, it can not be reset until global TZSC reset.
0: control register not locked
1: control register locked

5.5.2 GTZC_TZSC secure configuration register 1 (GTZC_TZSC_SECCFGR1)

Address offset: 0x010

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by privilege secure transaction when corresponding TZSC_PRIVCFGR register signal is set to 1. If a given PRIV bit is not set, the equivalent SEC bit can be written by non- privilege secure transaction.

Read accesses are authorized for any type of transactions, secure or not, privilege or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1SEC	TIM1SEC	COMPSEC	VREFBUFSEC	UCPD1SEC	USBFSSEC	FDCAN1SEC	LPTIM3SEC	LPTIM2SEC	I2C4SEC	LPUART1SEC	LPTIM1SEC	OPAMPSEC	DAC1SEC	CRSSEC	I2C3SEC
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
I2C2SEC	I2C1SEC	UART5SEC	UART4SEC	USART3SEC	USART2SEC	SPI3SEC	SPI2SEC	IWDGSEC	WWDGSEC	TIM7SEC	TIM6SEC	TIM5SEC	TIM4SEC	TIM3SEC	TIM2SEC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SPI1SEC**: secure access mode for SPI1
0: non-secure
1: secure

Bit 30 **TIM1SEC**: secure access mode for TIM1
0: non-secure
1: secure

Bit 29 **COMPSEC**: secure access mode for COMP
0: non-secure
1: secure

Bit 28 **VREFBUFSEC**: secure access mode for VREFBUF
0: non-secure
1: secure

Bit 27 **UCPD1SEC**: secure access mode for UCPD1
0: non-secure
1: secure

Bit 26 **USBFSSEC**: secure access mode for USB FS
0: non-secure
1: secure

Bit 25 **FDCAN1SEC**: secure access mode for FDCAN1
0: non-secure
1: secure

Bit 24 **LPTIM3SEC**: secure access mode for LPTIM3
0: non-secure
1: secure

Bit 23 **LPTIM2SEC**: secure access mode for LPTIM2
0: non-secure
1: secure

Bit 22 **I2C4SEC**: secure access mode for I2C4
0: non-secure
1: secure

Bit 21 **LPUART1SEC**: secure access mode for LPUART1
0: non-secure
1: secure

Bit 20 **LPTIM1SEC**: secure access mode for LPTIM1
0: non-secure
1: secure

Bit 19 **OPAMPSEC**: secure access mode for OPAMP
0: non-secure
1: secure

Bit 18 **DAC1SEC**: secure access mode for DAC1
0: non-secure
1: secure

Bit 17 **CRSSEC**: secure access mode for CRS
0: non-secure
1: secure

Bit 16 **I2C3SEC**: secure access mode for I2C3
0: non-secure
1: secure

Bit 15 **I2C2SEC**: secure access mode for I2C2
0: non-secure
1: secure

Bit 14 **I2C1SEC**: secure access mode for I2C1
0: non-secure
1: secure

Bit 13 **UART5SEC**: secure access mode for UART5
0: non-secure
1: secure

Bit 12 **UART4SEC**: secure access mode for UART4
0: non-secure
1: secure

Bit 11 **USART3SEC**: secure access mode for USART3
0: non-secure
1: secure

Bit 10 **USART2SEC**: secure access mode for USART2
0: non-secure
1: secure

Bit 9 **SPI3SEC**: secure access mode for SPI3
0: non-secure
1: secure

Bit 8 **SPI2SEC**: secure access mode for SPI2
0: non-secure
1: secure

Bit 7 **IWDGSEC**: secure access mode for IWDG
0: non-secure
1: secure

Bit 6 **WWDGSEC**: secure access mode for WWDG
0: non-secure
1: secure

Bit 5 **TIM7SEC**: secure access mode for TIM7
0: non-secure
1: secure

Bit 4 **TIM6SEC**: secure access mode for TIM6
0: non-secure
1: secure

Bit 3 **TIM5SEC**: secure access mode for TIM5
0: non-secure
1: secure

Bit 2 **TIM4SEC**: secure access mode for TIM4
0: non-secure
1: secure

Bit 1 **TIM3SEC**: secure access mode for TIM3
0: non-secure
1: secure

Bit 0 **TIM2SEC**: secure access mode for TIM2
0: non-secure
1: secure

5.5.3 GTZC_TZSC secure configuration register 2 (GTZC_TZSC_SECCFGR2)

Address offset: 0x014

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by privilege secure transaction when corresponding TZSC_PRIVCFGR register signal is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by non- privilege secure transaction.

Read accesses are authorized for any type of transactions, secure or not, privilege or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1_REGSEC	FMC_REGSEC	SDMMC1SEC
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PKASEC	RNGSEC	HASHSEC	AESSEC	ADCSEC	ICACHE_REGSEC	TSCSEC	CRCSEC	DFSDM1SEC	SAI2SEC	SAI1SEC	TIM17SEC	TIM16SEC	TIM15SEC	USART1SEC	TIM8SEC
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.

Bit 18 **OCTOSPI1_REGSEC**: secure access mode for OCTOSPI1 registers
0: non-secure
1: secure

Bit 17 **FMC_REGSEC**: secure access mode for FMC registers
0: non-secure
1: secure

Bit 16 **SDMMC1SEC**: secure access mode for SDMMC1
0: non-secure
1: secure

Bit 15 **PKASEC**: secure access mode for PKA
0: non-secure
1: secure

Bit 14 **RNGSEC**: secure access mode for RNG
0: non-secure
1: secure

Bit 13 **HASHSEC**: secure access mode for HASH
0: non-secure
1: secure

Bit 12 **AESSEC**: secure access mode for AES
0: non-secure
1: secure

Bit 11 **ADCSEC**: secure access mode for ADC
0: non-secure
1: secure

Bit 10 **ICACHE_REGSEC**: secure access mode for ICACHE registers
0: non-secure
1: secure

Bit 9 **TSCSEC**: secure access mode for TSC
0: non-secure
1: secure

Bit 8 **CRCSEC**: secure access mode for CRC
0: non-secure
1: secure

Bit 7 **DFSDM1SEC**: secure access mode for DFSDM1
0: non-secure
1: secure

Bit 6 **SAI2SEC**: secure access mode for SAI2
0: non-secure
1: secure

Bit 5 **SAI1SEC**: secure access mode for SAI1
0: non-secure
1: secure

Bit 4 **TIM17SEC**: secure access mode for TIM17
0: non-secure
1: secure

Bit 3 **TIM16SEC**: secure access mode for TIM16
0: non-secure
1: secure

Bit 2 **TIM15SEC**: secure access mode for TIM15
0: non-secure
1: secure

Bit 1 **USART1SEC**: secure access mode for USART1
0: non-secure
1: secure

Bit 0 **TIM8SEC**: secure access mode for TIM8
0: non-secure
1: secure

5.5.4 GTZC_TZSC privilege configuration register 1 (GTZC_TZSC_PRIVCFGR1)

Address offset: 0x020

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privilege transaction when corresponding TZSC_SECCFGR register signal is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by non-secure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privilege or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1PRIV	TIM1PRIV	COMPPRIV	VREFBUFPRIV	UCPD1PRIV	USBFSPRIV	FDCAN1PRIV	LPTIM3PRIV	LPTIM2PRIV	I2C4PRIV	LPUART1PRIV	LPTIM1PRIV	OPAMPPRIV	DAC1PRIV	CRSPRIV	I2C3PRIV
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
I2C2PRIV	I2C1PRIV	UART5PRIV	UART4PRIV	USART3PRIV	USART2PRIV	SPI3PRIV	SPI2PRIV	IWDGPRIV	WWDGPRIV	TIM7PRIV	TIM6PRIV	TIM5PRIV	TIM4PRIV	TIM3PRIV	TIM2PRIV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SPI1PRIV**: privilege access mode for SPI1
0: non-privilege
1: privilege

Bit 30 **TIM1PRIV**: privilege access mode for TIM1
0: non-privilege
1: privilege

Bit 29 **COMPPRIV**: privilege access mode for COMP
0: non-privilege
1: privilege

Bit 28 **VREFBUFPRIV**: privilege access mode for VREFBUF
0: non-privilege
1: privilege

Bit 27 **UCPD1PRIV**: privilege access mode for UCPD1
0: non-privilege
1: privilege

Bit 26 **USBFSPRIV**: privilege access mode for USB FS
0: non-privilege
1: privilege

Bit 25 **FDCAN1PRIV**: privilege access mode for FDCAN1
0: non-privilege
1: privilege

Bit 24 **LPTIM3PRIV**: privilege access mode for LPTIM3
0: non-privilege
1: privilege

Bit 23 **LPTIM2PRIV**: privilege access mode for LPTIM2
0: non-privilege
1: privilege

Bit 22 **I2C4PRIV**: privilege access mode for I2C4
0: non-privilege
1: privilege

Bit 21 **LPUART1PRIV**: privilege access mode for LPUART1
0: non-privilege
1: privilege

Bit 20 **LPTIM1PRIV**: privilege access mode for LPTIM1
0: non-privilege
1: privilege

Bit 19 **OPAMPPRIV**: privilege access mode for OPAMP
0: non-privilege
1: privilege

Bit 18 **DAC1PRIV**: privilege access mode for DAC1
0: non-privilege
1: privilege

Bit 17 **CRSPRIV**: privilege access mode for CRS
0: non-privilege
1: privilege

Bit 16 **I2C3PRIV**: privilege access mode for I2C3
0: non-privilege
1: privilege

Bit 15 **I2C2PRIV**: privilege access mode for I2C2
0: non-privilege
1: privilege

Bit 14 **I2C1PRIV**: privilege access mode for I2C1
0: non-privilege
1: privilege

Bit 13 **UART5PRIV**: privilege access mode for UART5
0: non-privilege
1: privilege

Bit 12 **UART4PRIV**: privilege access mode for UART4
0: non-privilege
1: privilege

Bit 11 **USART3PRIV**: privilege access mode for USART3
0: non-privilege
1: privilege

Bit 10 **USART2PRIV**: privilege access mode for USART2
0: non-privilege
1: privilege

Bit 9 **SPI3PRIV**: privilege access mode for SPI3
0: non-privilege
1: privilege

Bit 8 **SPI2PRIV**: privilege access mode for SPI2
0: non-privilege
1: privilege

Bit 7 **IWDGPRIV**: privilege access mode for IWDG
0: non-privilege
1: privilege

Bit 6 **WWDGPRIV**: privilege access mode for WWDG
0: non-privilege
1: privilege

Bit 5 **TIM7PRIV**: privilege access mode for TIM7
0: non-privilege
1: privilege

Bit 4 **TIM6PRIV**: privilege access mode for TIM6
0: non-privilege
1: privilege

Bit 3 **TIM5PRIV**: privilege access mode for TIM5
0: non-privilege
1: privilege

Bit 2 **TIM4PRIV**: privilege access mode for TIM4
0: non-privilege
1: privilege

Bit 1 **TIM3PRIV**: privilege access mode for TIM3
0: non-privilege
1: privilege

Bit 0 **TIM2PRIV**: privilege access mode for TIM2
0: non-privilege
1: privilege

5.5.5 GTZC_TZSC privilege configuration register 2 (GTZC_TZSC_PRIVCFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privilege transaction when corresponding TZSC_SECCFGR register signal is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by non-secure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privilege or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1_REGPRIV	FMC_REGPRIV	SDMMC1PRIV
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PKAPRIV	RNGPRIV	HASHPRIV	AESPRIV	ADCPRIV	ICACHE_REGPRIV	TSCPRIV	CRCPRIV	DFSDM1PRIV	SAI2PRIV	SAI1PRIV	TIM17PRIV	TIM16PRIV	TIM15PRIV	USART1PRIV	TIM8PRIV
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.

Bit 18 **OCTOSPI1_REGPRIV**: privilege access mode for OCTOSPI1 registers
0: non-privilege
1: privilege

Bit 17 **FMC_REGPRIV**: privilege access mode for FMC registers
0: non-privilege
1: privilege

Bit 16 **SDMMC1PRIV**: privilege access mode for SDMMC1
0: non-privilege
1: privilege

Bit 15 **PKAPRIV**: privilege access mode for PKA
0: non-privilege
1: privilege

Bit 14 **RNGPRIV**: privilege access mode for RNG
0: non-privilege
1: privilege

Bit 13 **HASHPRIV**: privilege access mode for HASH
0: non-privilege
1: privilege

Bit 12 **AESPRIV**: privilege access mode for AES
0: non-privilege
1: privilege

Bit 11 **ADCPRIV**: privilege access mode for ADC
0: non-privilege
1: privilege

Bit 10 **ICACHE_REGPRIV**: privilege access mode for ICACHE registers
0: non-privilege
1: privilege

Bit 9 **TSCPRIV**: privilege access mode for TSC
0: non-privilege
1: privilege

Bit 8 **CRCPRIV**: privilege access mode for CRC
0: non-privilege
1: privilege

Bit 7 **DFSDM1PRIV**: privilege access mode for DFSDM1
0: non-privilege
1: privilege

Bit 6 **SAI2PRIV**: privilege access mode for SAI2
0: non-privilege
1: privilege

Bit 5 **SAI1PRIV**: privilege access mode for SAI1
0: non-privilege
1: privilege

Bit 4 **TIM17PRIV**: privilege access mode for TIM17
0: non-privilege
1: privilege

Bit 3 **TIM16PRIV**: privilege access mode for TIM16
0: non-privilege
1: privilege

Bit 2 **TIM15PRIV**: privilege access mode for TIM15
0: non-privilege
1: privilege

Bit 1 **USART1PRIV**: privilege access mode for USART1
0: non-privilege
1: privilege

Bit 0 **TIM8PRIV**: privilege access mode for TIM8
0: non-privilege
1: privilege

5.5.6 GTZC_TZSC external memory x non-secure watermark register 1 (GTZC_TZSC_MPCWMxANSR)

Address offset: 0x030 + 0x008 * (x-1), (x = 1 to 3)

Reset value: 0x0000 0000

The given reset value is valid when TZEN = 1. The reset value is 0x0800 0000 when TZEN = 0.

Secure access only.

Caution: When NSWM1STRT + NSWM1LGTH is higher than the maximum size allowed for the memory, a saturation of NSWM1LGTH is applied automatically.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	NSWM1LGTH[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.	Res.	NSWM1STRT[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **NSWM1LGTH[11:0]**: length of the first non-secure area (multiple of 128 Kbytes)

Note: If programmed NSWM1LGTH + NSWM1STRT is over 2048, the value stored in the register is truncated to (0x800 - NSWM1STRT). Any subsequent read returns this value.

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

Bits 10:0 **NSWM1STRT[10:0]**: offset address for the first non-secure area (multiple of 128 Kbytes)

Note: External memories which are watermark controlled start fully non-secure at reset when TZEN = 0. When TZEN = 1, external memories start fully secure (inverted reset-value).

5.5.7 GTZC_TZSC external memory x non-secure watermark register 2 (GTZC_TZSC_MPCWMxBNSR)

Address offset: 0x034 + 0x008 * (x-1), (x = 1 to 2)

Reset value: 0x0000 0000

The given reset value is valid when TZEN = 1. The reset value is 0x0800 0000 when TZEN = 0.

Secure access only.

Caution: When NSWM1STRT + NSWM1LGTH is higher than the maximum size allowed for the memory, a saturation of NSWM1LGTH is applied automatically.

Note: If NSWM1LGTH = 0, the region 1 is disabled and the only non-secure memory space is defined by NSWMPxWM2.

If both NSWMPxWM1 and NSWMPxWM2 have the reset value 0x0000 0000, all the memory space of the external memory x (FMC NOR/SRAM or OCTOSPI) is secure.

If NSWM1LGTH = 0x800 and NSWM1STRT = 0, the whole 256-Mbyte memory space is non-secure (independent of NSWMPxWM2 value).

If NSWM1LGTH = 0x001 and NSWM1STRT = 0x7FF, only one 128-Kbyte block is defined as non-secure (at address offset = 0x0FFE 0000, ending at 0x0FFF FFFF).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	NSWM2LGTH[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.	Res.	NSWM2STRT[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **NSWM2LGTH[11:0]**: length of the second non-secure area (multiple of 128 Kbytes)

Note: If programmed NSWM2LGTH + NSWM2STRT is over 2048, the value stored in the register is truncated to (0x800 - NSWM2STRT). Any subsequent read returns this value.

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

Bits 10:0 **NSWM2STRT[10:0]**: offset address for the second non-secure area (multiple of 128 Kbytes)

Note: External memories which are watermark controlled start fully non-secure at reset when TZEN = 0. When TZEN = 1, external memories start fully secure (inverted reset-value).

5.5.8 GTZC_TZSC register map and reset values

Table 26. GTZC_TZSC 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	GTZC_TZSC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
	Reset value																																0
0x004 to 0x00C	Reserved	Reserved																															
0x010	GTZC_TZSC_SECCFGR1	SPI1SEC	TIM1SEC	COMPSEC	VREFBUFSEC	UCPD1SEC	USBFSSEC	FDCAN1SEC	LPTIM3SEC	LPTIM2SEC	I2C4SEC	LPUART1SEC	LPTIM1SEC	OPAMPSEC	DAC1SEC	CRSSEC	I2C3SEC	I2C2SEC	I2C1SEC	UART5SEC	UART4SEC	USART3SEC	USART2SEC	SPI3SEC	SPI2SEC	IWDGSEC	WWDGSEC	TIM7SEC	TIM6SEC	TIM5SEC	TIM4SEC	TIM3SEC	TIM2SEC
	Reset value	0	0	0	0	0	0	0	0	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	GTZC_TZSC_SECCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1_REGSEC	FMC_REGSEC	SDMMC1SEC	PKASEC	RNGSEC	HASHSEC	AESSEC	ADCSEC	ICACHE_REGSEC	TSCSEC	CRCSEC	DFSDM1SEC	SAI2SEC	SAI1SEC	TIM17SEC	TIM16SEC	TIM15SEC	USART1SEC	TIM8SEC
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018 to 0x01C	Reserved	Reserved																															

Table 26. GTZC_TZSC 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	GTZC_TZSC _PRIVCFGR1	SPI1PRIV	TIM1PRIV	COMPPRIV	VREFBUFPRIV	UCPD1PRIV	USBFSPRIV	FDCAN1PRIV	LPTIM3PRIV	LPTIM2PRIV	I2C4PRIV	LPUART1PRIV	LPTIM1PRIV	OPAMPPRIV	DAC1PRIV	CRSPRIV	I2C3PRIV	I2C2PRIV	I2C1PRIV	UART5PRIV	UART4PRIV	USART3PRIV	USART2PRIV	SPI3PRIV	SPI2PRIV	IWDGPRIV	WWDGPRIV	TIM7PRIV	TIM6PRIV	TIM5PRIV	TIM4PRIV	TIM3PRIV	TIM2PRIV
	Reset value	0	0	0	0	0	0	0	0	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	GTZC_TZSC _PRIVCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1_REGPRIV	FMC_REGPRIV	SDMMC1PRIV	PKAPRIV	RNGPRIV	HASHPRIV	AESPRIV	ADCPRIV	ICACHE_REGPRIV	TSCPRIV	CRCPRIV	DFSDM1PRIV	SAI2PRIV	SAI1PRIV	TIM17PRIV	TIM16PRIV	TIM15PRIV	USART1PRIV
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028 to 0x02C	Reserved	Reserved																															
0x030	GTZC_TZSC _MPCWM1ANSR	Res.	Res.	Res.	Res.	NSWM1LGTH[11:0]												Res.	Res.	Res.	Res.	Res.	NSWM1STRT[10:0]										
	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
0x034	GTZC_TZSC _MPCWM1BNSR	Res.	Res.	Res.	Res.	NSWM2LGTH[11:0]												Res.	Res.	Res.	Res.	Res.	NSWM2STRT[10:0]										
	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
0x038	GTZC_TZSC _MPCWM2ANSR	Res.	Res.	Res.	Res.	NSWM1LGTH[11:0]												Res.	Res.	Res.	Res.	Res.	NSWM1STRT[10:0]										
	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
0x03C	GTZC_TZSC _MPCWM2BNSR	Res.	Res.	Res.	Res.	NSWM2LGTH[11:0]												Res.	Res.	Res.	Res.	Res.	NSWM2STRT[10:0]										
	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
0x040	GTZC_TZSC _MPCWM3ANSR	Res.	Res.	Res.	Res.	NSWM1LGTH[11:0]												Res.	Res.	Res.	Res.	Res.	NSWM1STRT[10:0]										
	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

Refer to *Section 2.3 on page 87* for the register boundary addresses.

5.6 GTZC_MPCBB registers

All registers are accessed only by words (32-bit).

5.6.1 GTZC_MPCBBx control register (GTZC_MPCBBx_CR) (x = 1 to 2)

Address offset: 0x800 + 0x400 * (x - 1)

Reset value: 0x0000 0000

Secure access only.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
															rw

Bit 31 **SRWILADIS**: secure read/write illegal access disable

This bit disables the detection of an illegal access when a secure read/write transaction access a non-secure blocks of the block-based SRAM (secure fetch on non-secure block is always considered illegal).

0: enabled, secure read/write acces not allowed on non-secure SRAM block

1: disabled, secure read/write access allowed on non-secure SRAM block

Bit 30 **INVSECSTATE**: default security state

This bit is used to invert the MPCBB status information (secure or non-secure) connected to the RCC, in order to define the MPCBB clock control as secure or not.

0: default state (source clock secured if a secure area exists in the MPCBB and vice-versa)

1: invert the state, source clock remains secure even if no secure block is set in the MPCBB

Bits 29:1 Reserved, must be kept at reset value.

Bit 0 **LCK**: lock the control register of the MPCBB sub-block until next reset

This bit is unset by default and once set, it can not be reset until global TZSC reset.

0: control register not locked

1: control register locked

5.6.2 GTZC_MPCBB1 lock register 1(GTZC_MPCBB1_LCKVTR1)

Address offset: 0x810

Reset value: 0x0000 0000

Secure access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKSB23	LCKSB22	LCKSB21	LCKSB20	LCKSB19	LCKSB18	LCKSB17	LCKSB16
								rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCKSB15	LCKSB14	LCKSB13	LCKSB12	LCKSB11	LCKSB10	LCKSB9	LCKSB8	LCKSB7	LCKSB6	LCKSB5	LCKSB4	LCKSB3	LCKSB2	LCKSB1	LCKSB0
rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo

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

Bits 23:0 **LCKSB[23:0]**: lock/unlock status of secure access mode for the super-blocks 0 to 23
0x0000 0000: security configuration unlocked for all super-blocks
....
0x0000 00FF: security configuration locked **only** for super-blocks 0 to 7
.....
0x0080 0001: security configuration locked for super-blocks 0 and 23

5.6.3 GTZC_MPCBB2 lock register 1 (GTZC_MPCBB2_LCKVTR1)

Address offset: 0xC10

Reset value: 0x0000 0000

Secure access only.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKSB7	LCKSB6	LCKSB5	LCKSB4	LCKSB3	LCKSB2	LCKSB1	LCKSB0
								rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo

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

Bits 7:0 **LCKSB[7:0]**: lock/unlock status of secure access mode for the super-blocks 0 to 7
0x00: security configuration unlocked for all super-blocks
....
0xFF: security configuration locked for super-blocks 0 to 7

5.6.4 GTZC_MPCBBx vector register y (GTZC_MPCBBx_VCTRy) (x = 1 to 2)

Address offset: 0x900 + 0x04 * y, (y = 0 to 23)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 1. The reset value is 0x0000 0000 when TZEN = 0.

Secure access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
B(31 + 32 * y)	B(30 + 32 * y)	B(29 + 32 * y)	B(28 + 32 * y)	B(27 + 32 * y)	B(26 + 32 * y)	B(25 + 32 * y)	B(24 + 32 * y)	B(23 + 32 * y)	B(22 + 32 * y)	B(21 + 32 * y)	B(20 + 32 * y)	B(19 + 32 * y)	B(18 + 32 * y)	B(17 + 32 * y)	B(16 + 32 * y)
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
B(15 + 32 * y)	B(14 + 32 * y)	B(13 + 32 * y)	B(12 + 32 * y)	B(11 + 32 * y)	B(10 + 32 * y)	B(9 + 32 * y)	B(8 + 32 * y)	B(7 + 32 * y)	B(6 + 32 * y)	B(5 + 32 * y)	B(4 + 32 * y)	B(3 + 32 * y)	B(2 + 32 * y)	B(1 + 32 * y)	B(32 * y)
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **B[31+ 32 * y:32 * y]**: define secure access mode for the super-block y

0x0000 0000: all blocks of super-block y are non-secure.

....

0x0000 00FF: **only** blocks 0 to 7 of super-block y are secure.

.....

0x8000 0001: **only** blocks 0 and 31 of super-block y are secure.

....

0xFFFF FFFF: all super-blocks are secure.

5.6.5 GTZC_MPCBB1 register map and reset values

Table 27. GTZC_MPCBB1 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
0x800	GTZC_MPCBB1 _CR	SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
	Reset value	0	0																														0
0x804 to 0x80C	Reserved	Reserved																															
0x810	GTZC_MPCBB1 _LCKVTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKSB23	LCKSB22	LCKSB21	LCKSB20	LCKSB19	LCKSB18	LCKSB17	LCKSB16	LCKSB15	LCKSB14	LCKSB13	LCKSB12	LCKSB11	LCKSB10	LCKSB9	LCKSB8	LCKSB7	LCKSB6	LCKSB5	LCKSB4	LCKSB3	LCKSB2	LCKSB1	LCKSB0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x814 to 0x8FC	Reserved	Reserved																															
0x900 + 0x004 *y (y = 0 to 23)	GTZC_MPCBB1 _VCTRy	B(31 + 32 * y)	B(30 + 32 * y)	B(29 + 32 * y)	B(28 + 32 * y)	B(27 + 32 * y)	B(26 + 32 * y)	B(25 + 32 * y)	B(24 + 32 * y)	B(23 + 32 * y)	B(22 + 32 * y)	B(21 + 32 * y)	B(20 + 32 * y)	B(19 + 32 * y)	B(18 + 32 * y)	B(17 + 32 * y)	B(16 + 32 * y)	B(15 + 32 * y)	B(14 + 32 * y)	B(13 + 32 * y)	B(12 + 32 * y)	B(11 + 32 * y)	B(10 + 32 * y)	B(9 + 32 * y)	B(8 + 32 * y)	B(7 + 32 * y)	B(6 + 32 * y)	B(5 + 32 * y)	B(4 + 32 * y)	B(3 + 32 * y)	B(2 + 32 * y)	B(1 + 32 * y)	B(32 * y)
	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

Refer to *Section 2.3 on page 87* for the register boundary addresses.

5.6.6 GTZC_MPCBB2 register map and reset values

Table 28. GTZC_MPCBB2 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
0xC00	GTZC_MPCBB2 _CR	SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
	Reset value	0	0																														0
0xC04 to 0xC0C	Reserved	Reserved																															
0xC10	GTZC_MPCBB2 _LCKVTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKSB7	LCKSB6	LCKSB5	LCKSB4	LCKSB3	LCKSB2	LCKSB1	LCKSB0
	Reset value																									0	0	0	0	0	0	0	0
0xC14 to 0xCFC	Reserved	Reserved																															
0xD00 + 0x04 *y (y =0 to 7)	GTZC_MPCBB2 _VCTRy	B(31 + 32 * y)	B(30 + 32 * y)	B(29 + 32 * y)	B(28 + 32 * y)	B(27 + 32 * y)	B(26 + 32 * y)	B(25 + 32 * y)	B(24 + 32 * y)	B(23 + 32 * y)	B(22 + 32 * y)	B(21 + 32 * y)	B(20 + 32 * y)	B(19 + 32 * y)	B(18 + 32 * y)	B(17 + 32 * y)	B(16 + 32 * y)	B(15 + 32 * y)	B(14 + 32 * y)	B(13 + 32 * y)	B(12 + 32 * y)	B(11 + 32 * y)	B(10 + 32 * y)	B(9 + 32 * y)	B(8 + 32 * y)	B(7 + 32 * y)	B(6 + 32 * y)	B(5 + 32 * y)	B(4 + 32 * y)	B(3 + 32 * y)	B(2 + 32 * y)	B(1 + 32 * y)	B(32 * y)
	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

Refer to *Section 2.3 on page 87* for the register boundary addresses.

5.7 GTZC_TZIC registers

All registers are accessed only by words (32-bit).

5.7.1 GTZC_TZIC interrupt enable register 1 (GTZC_TZIC_IER1)

Address offset: 0x400

Reset value: 0x0000 0000

Secure access only.

This register is used to enable/disable generation of GTZC_IRQn interrupt towards NVIC on illegal access event for each source.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1IE	TIM1IE	COMPIE	VREFBUFIE	UCPD1IE	USBFSIE	FDCAN1IE	LPTIM3IE	LPTIM2IE	I2C4IE	LPUART1IE	LPTIM1IE	OPAMPIE	DAC1IE	CRSIE	I2C3IE
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
I2C2IE	I2C1IE	UART5IE	UART4IE	USART3IE	USART2IE	SPI3IE	SPI2IE	IWDGIE	WWDGIE	TIM7IE	TIM6IE	TIM5IE	TIM4IE	TIM3IE	TIM2IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SPI1IE**: illegal access interrupt enable for SPI1
0: disabled
1: enabled

Bit 30 **TIM1IE**: illegal access interrupt enable for TIM1
0: disabled
1: enabled

Bit 29 **COMPIE**: illegal access interrupt enable for COMP
0: disabled
1: enabled

Bit 28 **VREFBUFIE**: illegal access interrupt enable for VREFBUF
0: disabled
1: enabled

Bit 27 **UCPD1IE**: illegal access interrupt enable for UCPD1
0: disabled
1: enabled

Bit 26 **USBFSIE**: illegal access interrupt enable for USBFS
0: disabled
1: enabled

Bit 25 **FDCAN1IE**: illegal access interrupt enable for FDCAN1
0: disabled
1: enabled

Bit 24 **LPTIM3IE**: illegal access interrupt enable for LPTIM3
0: disabled
1: enabled

Bit 23 **LPTIM2IE**:illegal access interrupt enable for LPTIM2
0: disabled
1: enabled

Bit 22 **I2C4IE**: illegal access interrupt enable for I2C4
0: disabled
1: enabled

Bit 21 **LPUART1IE**: illegal access interrupt enable for LPUART1
0: disabled
1: enabled

Bit 20 **LPTIM1IE**: illegal access interrupt enable for LPTIM1
0: disabled
1: enabled

Bit 19 **OPAMPIE**: illegal access interrupt enable for OPAMP
0: disabled
1: enabled

Bit 18 **DAC1IE**: illegal access interrupt enable for DAC1
0: disabled
1: enabled

Bit 17 **CRSIE**: illegal access interrupt enable for CRS
0: disabled
1: enabled

Bit 16 **I2C3IE**: illegal access interrupt enable for I2C3
0: disabled
1: enabled

Bit 15 **I2C2IE**: illegal access interrupt enable for I2C2
0: disabled
1: enabled

Bit 14 **I2C1IE**: illegal access interrupt enable for I2C1
0: disabled
1: enabled

Bit 13 **UART5IE**: illegal access interrupt enable for UART5
0: disabled
1: enabled

Bit 12 **UART4IE**: illegal access interrupt enable for UART4
0: disabled
1: enabled

Bit 11 **USART3IE**: illegal access interrupt enable for USART3
0: disabled
1: enabled

Bit 10 **USART2IE**: illegal access interrupt enable for USART2
0: disabled
1: enabled

Bit 9 **SPI3IE**: illegal access interrupt enable for SPI3
0: disabled
1: enabled

Bit 8 **SPI2IE**: illegal access interrupt enable for SPI2
0: disabled
1: enabled

Bit 7 **IWDGIE**: illegal access interrupt enable for IWDG
0: disabled
1: enabled

Bit 6 **WWDGIE**: illegal access interrupt enable for WWDG
0: disabled
1: enabled

Bit 5 **TIM7IE**: illegal access interrupt enable for TIM7
0: disabled
1: enabled

Bit 4 **TIM6IE**: illegal access interrupt enable for TIM6
0: disabled
1: enabled

Bit 3 **TIM5IE**: illegal access interrupt enable for TIM5
0: disabled
1: enabled

Bit 2 **TIM4IE**: illegal access interrupt enable for TIM4
0: disabled
1: enabled

Bit 1 **TIM3IE**: illegal access interrupt enable for TIM3
0: disabled
1: enabled

Bit 0 **TIM2IE**: illegal access interrupt enable for TIM2
0: disabled
1: enabled

5.7.2 GTZC_TZIC interrupt enable register 2 (GTZC_TZIC_IER2)

Address offset: 0x404

Reset value: 0x0000 0000

Secure access only.

This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	OTFDEC1IE	EXTIIE	FLASH_REGIE	FLASHIE	RCCIE	DMAMUX1IE	DMA2IE	DMA1IE	SYSCFGIE	PWRIE	RTCIE	OCTOSPI1_REGIE	FMC_REGIE	SDMMC1IE
		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
PKAIE	RNGIE	HASHIE	AESIE	ADCIE	ICACHE_REGIE	TSCIE	CRCIE	DFSDM1IE	SAI2IE	SAI1IE	TIM17IE	TIM16IE	TIM15IE	USART1IE	TIM8IE
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 **OTFDEC1IE**: illegal access interrupt enable for OTFDEC
- 0: disabled
- 1: enabled

Bit 28 **EXTIIE**: illegal access interrupt enable for EXTI
- 0: disabled
- 1: enabled

Bit 27 **FLASH_REGIE**: illegal access interrupt enable for FLASH registers
- 0: disabled
- 1: enabled

Bit 26 **FLASHIE**: illegal access interrupt enable for FLASH
- 0: disabled
- 1: enabled

Bit 25 **RCCIE**: illegal access interrupt enable for RCC
- 0: disabled
- 1: enabled

Bit 24 **DMAMUX1IE**: illegal access interrupt enable for DMAMUX1
- 0: disabled
- 1: enabled

Bit 23 **DMA2IE**: illegal access interrupt enable for DMA2
- 0: disabled
- 1: enabled

Bit 22 **DMA1IE**: illegal access interrupt enable for DMA1
0: disabled
1: enabled

Bit 21 **SYSCFGIE**: illegal access interrupt enable for SYSCFG
0: disabled
1: enabled

Bit 20 **PWRIE**: illegal access interrupt enable for PWR
0: disabled
1: enabled

Bit 19 **RTCIE**: illegal access interrupt enable for RTC
0: disabled
1: enabled

Bit 18 **OCTOSPI1_REGIE**: illegal access interrupt enable for OCTOSPI1 registers
0: disabled
1: enabled

Bit 17 **FMC_REGIE**: illegal access interrupt enable for FMC registers
0: disabled
1: enabled

Bit 16 **SDMMC1IE**: illegal access interrupt enable for SDMMC1
0: disabled
1: enabled

Bit 15 **PKAIE:** illegal access interrupt enable for PKA
0: disabled
1: enabled

Bit 14 **RNGIE**: illegal access interrupt enable for RNG
0: disabled
1: enabled

Bit 13 **HASHIE**: illegal access interrupt enable for HASH
0: disabled
1: enabled

Bit 12 **AESIE**: illegal access interrupt enable for AES
0: disabled
1: enabled

Bit 11 **ADCIE**: illegal access interrupt enable for ADC
0: disabled
1: enabled

Bit 10 **ICACHE_REGIE**: illegal access interrupt enable for ICACHE registers
0: disabled
1: enabled

Bit 9 **TSCIE**: illegal access interrupt enable for TSC
0: disabled
1: enabled

Bit 8 **CRCIE**: illegal access interrupt enable for CRC
0: disabled
1: enabled

Bit 7 **DFSDM1IE**: illegal access interrupt enable for DFSDM1
0: disabled
1: enabled

Bit 6 **SAI2IE**: illegal access interrupt enable for SAI2
0: disabled
1: enabled

Bit 5 **SAI1IE**: illegal access interrupt enable for SAI1
0: disabled
1: enabled

Bit 4 **TIM17IE**: illegal access interrupt enable for TIM17
0: disabled
1: enabled

Bit 3 **TIM16IE**: illegal access interrupt enable for TIM16
0: disabled
1: enabled

Bit 2 **TIM15IE**: illegal access interrupt enable for TIM15
0: disabled
1: enabled

Bit 1 **USART1IE**: illegal access interrupt enable for USART1
0: disabled
1: enabled

Bit 0 **TIM8IE**: illegal access interrupt enable forTIM8
0: disabled
1: enabled

5.7.3 GTZC_TZIC interrupt enable register 3 (GTZC_TZIC_IER3)

Address offset: 0x408

Reset value: 0x0000 0000

Secure access only.

This register is used to enable interrupt of illegal 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.	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	OCTOSPI1_MEMIE	FMC_MEMIE	TZICIE	TZSCIE
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 7 **MPCBB2_REGIE**: illegal access interrupt enable for MPCBB2 registers
0: disabled
1: enabled

Bit 6 **SRAM2IE**: illegal access interrupt enable for SRAM2
0: disabled
1: enabled

Bit 5 **MPCBB1_REGIE**: illegal access interrupt enable for MPCBB1 registers
0: disabled
1: enabled

Bit 4 **SRAM1IE**: illegal access interrupt enable for SRAM1
0: disabled
1: enabled

Bit 3 **OCTOSPI1_MEMIE**: illegal access interrupt enable for OCTOSPI1 memory interface
0: disabled
1: enabled

Bit 2 **FMC_MEMIE**: illegal access interrupt enable for FMC NAND and FMC NOR memories
0: disabled
1: enabled

Bit 1 **TZICIE**: illegal access interrupt enable for TZIC registers
0: disabled
1: enabled

Bit 0 **TZSCIE**: illegal access interrupt enable TZSC
0: disabled
1: enabled

5.7.4 GTZC_TZIC status register 1 (GTZC_TZIC_SR1)

Address offset: 0x410

Reset value: 0x0000 0000

Secure access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1F	TIM1F	COMPF	VREFBUFF	UCPD1F	USBFSF	FDCAN1F	LPTIM3F	LPTIM2F	I2C4F	LPUART1F	LPTIM1F	OPAMPF	DAC1F	CRSF	I2C3F
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
I2C2F	I2C1F	UART5F	UART4F	USART3F	USART2F	SPI3F	SPI2F	IWDGF	WWDGF	TIM7F	TIM6F	TIM5F	TIM4F	TIM3F	TIM2F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **SPI1F**: illegal access flag for SPI1
0: no illegal access event
1: an illegal access event pending

Bit 30 **TIM1F**: illegal access flag for TIM1
0: no illegal access event
1: an illegal access event pending

Bit 29 **COMPF**: illegal access flag for COMP
0: no illegal access event
1: an illegal access event pending

Bit 28 **VREFBUFF**: illegal access flag for VREFBUF
0: no illegal access event
1: an illegal access event pending

Bit 27 **UCPD1F**: illegal access flag for UCPD1
0: no illegal access event
1: an illegal access event pending

Bit 26 **USBFSF**: illegal access flag for USB FS
0: no illegal access event
1: an illegal access event pending

Bit 25 **FDCAN1F**: illegal access flag for FDCAN1
0: no illegal access event
1: an illegal access event pending

Bit 24 **LPTIM3F**: illegal access flag for LPTIM3
0: no illegal access event
1: an illegal access event pending

Bit 23 **LPTIM2F**: illegal access flag for LPTIM2
0: no illegal access event
1: an illegal access event pending

Bit 22 **I2C4F**: illegal access flag for I2C4
0: no illegal access event
1: an illegal access event pending

Bit 21 **LPUART1F**: illegal access flag for LPUART1
0: no illegal access event
1: an illegal access event pending

Bit 20 **LPTIM1F**: illegal access flag for LPTIM1
0: no illegal access event
1: an illegal access event pending

Bit 19 **OPAMPF**: illegal access flag for OPAMP
0: no illegal access event
1: an illegal access event pending

Bit 18 **DAC1F**: illegal access flag for DAC1
0: no illegal access event
1: an illegal access event pending

Bit 17 **CRSF**: illegal access flag for CRS
0: no illegal access event
1: an illegal access event pending

Bit 16 **I2C3F**: illegal access flag for I2C3
0: no illegal access event
1: an illegal access event pending

Bit 15 **I2C2F**: illegal access flag for I2C2
0: no illegal access event
1: an illegal access event pending

Bit 14 **I2C1F**: illegal access flag for I2C1
0: no illegal access event
1: an illegal access event pending

Bit 13 **UART5F**: illegal access flag for UART5
0: no illegal access event
1: an illegal access event pending

Bit 12 **UART4F**: illegal access flag for UART4
0: no illegal access event
1: an illegal access event pending

Bit 11 **USART3F**: illegal access flag for USART3
0: no illegal access event
1: an illegal access event pending

Bit 10 **USART2F**: illegal access flag for USART2
0: no illegal access event
1: an illegal access event pending

Bit 9 **SPI3F**: illegal access flag for SPI3
0: no illegal access event
1: an illegal access event pending

Bit 8 **SPI2F**: illegal access flag for SPI2
0: no illegal access event
1: an illegal access event pending

Bit 7 **IWDGF**: illegal access flag for IWDG
0: no illegal access event
1: an illegal access event pending

Bit 6 **WWDGF**: illegal access flag for WWDG
0: no illegal access event
1: an illegal access event pending

Bit 5 **TIM7F**: illegal access flag for TIM7
0: no illegal access event
1: an illegal access event pending

Bit 4 **TIM6F**: illegal access flag for TIM6
0: no illegal access event
1: an illegal access event pending

Bit 3 **TIM5F**: illegal access flag for TIM5
0: no illegal access event
1: an illegal access event pending

Bit 2 **TIM4F**: illegal access flag for TIM4
0: no illegal access event
1: an illegal access event pending

Bit 1 **TIM3F**: illegal access flag for TIM3
0: no illegal access event
1: an illegal access event pending

Bit 0 **TIM2F**: illegal access flag for TIM2
0: no illegal access event
1: an illegal access event pending

5.7.5 GTZC_TZIC status register 2 (GTZC_TZIC_SR2)

Address offset: 0x414

Reset value: 0x0000 0000

Secure access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	OTFDEC1F	EXTIF	FLASH_REGF	FLASHF	RCCF	DMAMUX1F	DMA2F	DMA1F	SYSCFGF	PWRF	RTCF	OCTOSPI1_REGF	FMC_REGF	SDMMC1F
		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
PKAF	RNGF	HASHF	AESF	ADCF	ICACHE_REGF	TSCF	CRCF	DFSDM1F	SAI2F	SAI1F	TIM17F	TIM16F	TIM15F	USART1F	TIM8F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **OTFDEC1F**: illegal access flag for OTFDEC1
0: no illegal access event
1: an illegal access event pending

Bit 28 **EXTIF**: illegal access flag for EXTI
0: no illegal access event
1: an illegal access event pending

Bit 27 **FLASH_REGF**: illegal access flag for FLASH registers
0: no illegal access event
1: an illegal access event pending

Bit 26 **FLASHF**: illegal access flag for FLASH
0: no illegal access event
1: an illegal access event pending

Bit 25 **RCCF**: illegal access flag for RCC
0: no illegal access event
1: an illegal access event pending

Bit 24 **DMAMUX1F**: illegal access flag for DMAMUX1
0: no illegal access event
1: an illegal access event pending

Bit 23 **DMA2F**: illegal access flag for DMA2
0: no illegal access event
1: an illegal access event pending

Bit 22 **DMA1F**: illegal access flag for DMA1
0: no illegal access event
1: an illegal access event pending

Bit 21 **SYSCFGF**: illegal access flag for SYSCFG
0: no illegal access event
1: an illegal access event pending

Bit 20 **PWRF**: illegal access flag for PWR
0: no illegal access event
1: an illegal access event pending

Bit 19 **RTCF**: illegal access flag for RTC
0: no illegal access event
1: an illegal access event pending

Bit 18 **OCTOSPI1_REGF**: illegal access flag for OCTOSPI1 registers
0: no illegal access event
1: an illegal access event pending

Bit 17 **FMC_REGF**: illegal access flag for FMC registers
0: no illegal access event
1: an illegal access event pending

Bit 16 **SDMMC1F**: illegal access flag for SDMMC1
0: no illegal access event
1: an illegal access event pending

Bit 15 **PKAF**: illegal access flag for PKA
0: no illegal access event
1: an illegal access event pending

Bit 14 **RNGF**: illegal access flag for RNG
0: no illegal access event
1: an illegal access event pending

Bit 13 **HASHF**: illegal access flag for HASH
0: no illegal access event
1: an illegal access event pending

Bit 12 **AESF**: illegal access flag for AES
0: no illegal access event
1: an illegal access event pending

Bit 11 **ADCF**: illegal access flag for ADC
0: no illegal access event
1: an illegal access event pending

Bit 10 **ICACHE_REGF**: illegal access flag for ICACHE registers
0: no illegal access event
1: an illegal access event pending

Bit 9 **TSCF**: illegal access flag for TSC
0: no illegal access event
1: an illegal access event pending

Bit 8 **CRCF**: illegal access flag for CRC
0: no illegal access event
1: an illegal access event pending

Bit 7 **DFSDM1F**: illegal access flag for DFSDM1
0: no illegal access event
1: an illegal access event pending

Bit 6 **SAI2F**: illegal access flag for SAI2
0: no illegal access event
1: an illegal access event pending

Bit 5 **SAI1F**: illegal access flag for SAI1
0: no illegal access event
1: an illegal access event pending

Bit 4 **TIM17F**: illegal access flag for TIM17
0: no illegal access event
1: an illegal access event pending

Bit 3 **TIM16F**: illegal access flag for TIM16
0: no illegal access event
1: an illegal access event pending

Bit 2 **TIM15F**: illegal access flag for TIM15
0: no illegal access event
1: an illegal access event pending

Bit 1 **USART1F**: illegal access flag for USART1
0: no illegal access event
1: an illegal access event pending

Bit 0 **TIM8F**: illegal access flag for TIM8
0: no illegal access event
1: an illegal access event pending

5.7.6 GTZC_TZIC status register 3 (GTZC_TZIC_SR3)

Address offset: 0x418

Reset value: 0x0000 0000

Secure access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	OCTOSPI1_MEMF	FMC_MEMF	TZICF	TZSCF
								r	r	r	r	r	r	r	r

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

Bit 7 **MPCBB2_REGF**: illegal access flag for MPCBB2 registers
0: no illegal access event
1: an illegal access event pending

Bit 6 **SRAM2F**: illegal access flag for SRAM2
0: no illegal access event
1: an illegal access event pending

Bit 5 **MPCBB1_REGF**: illegal access flag for MPCBB1 registers
0: no illegal access event
1: an illegal access event pending

Bit 4 **SRAM1F**: illegal access flag for SRAM1
0: no illegal access event
1: an illegal access event pending

Bit 3 **OCTOSPI1_MEMF**: illegal access flag for OCTOSPI memory interface
0: no illegal access event
1: an illegal access event pending

Bit 2 **FMC_MEMF**: illegal access flag for FMC NAND and FMC NOR memory interface
0: no illegal access event
1: an illegal access event pending

Bit 1 **TZICF**: illegal access flag for TZIC
0: no illegal access event
1: an illegal access event pending

Bit 0 **TZSCF**: illegal access flag for TZSC
0: no illegal access event
1: an illegal access event pending

5.7.7 GTZC_TZIC flag clear register 1 (GTZC_TZIC_FCR1)

Address offset: 0x420

Reset value: 0x0000 0000

Secure access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1FC	TIM1FC	COMPFC	VREFBUFFC	UCPD1FC	USBFSFC	FDCAN1FC	LPTIM3FC	LPTIM2FC	I2C4FC	LPUART1FC	LPTIM1FC	OPAMPFC	DAC1FC	CRSFC	I2C3FC
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
I2C2FC	I2C1FC	UART5FC	UART4FC	USART3FC	USART2FC	SPI3FC	SPI2FC	IWDGFC	WWDGFC	TIM7FC	TIM6FC	TIM5FC	TIM4FC	TIM3FC	TIM2FC
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **SPI1FC**: clear the illegal access flag for SPI1
0: no action
1: status flag cleared

Bit 30 **TIM1FC**: clear the illegal access flag for TIM1
0: no action
1: status flag cleared

Bit 29 **COMPFC**: clear the illegal access flag for COMP
0: no action
1: status flag cleared

Bit 28 **VREFBUFFC**: clear the illegal access flag for VREFBUF
0: no action
1: status flag cleared

Bit 27 **UCPD1FC**: clear the illegal access flag for UCPD1
0: no action
1: status flag cleared

Bit 26 **USBFSFC**: clear the illegal access flag for USB FS
0: no action
1: status flag cleared

Bit 25 **FDCAN1FC**: clear the illegal access flag for FDCAN1
0: no action
1: status flag cleared

Bit 24 **LPTIM3FC**: clear the illegal access flag for LPTIM3
0: no action
1: status flag cleared

Bit 23 **LPTIM2FC**: clear the illegal access flag for LPTIM2
0: no action
1: status flag cleared

Bit 22 **I2C4FC**: clear the illegal access flag for I2C4
0: no action
1: status flag cleared

Bit 21 **LPUART1FC**: clear the illegal access flag for LPUART1
0: no action
1: status flag cleared

Bit 20 **LPTIM1FC**: clear the illegal access flag for LPTIM1
0: no action
1: status flag cleared

Bit 19 **OPAMPFC**: clear the illegal access flag for OPAMP
0: no action
1: status flag cleared

Bit 18 **DAC1FC**: clear the illegal access flag for DAC1
0: no action
1: status flag cleared

Bit 17 **CRSFC**: clear the illegal access flag for CRS
0: no action
1: status flag cleared

Bit 16 **I2C3FC**: clear the illegal access flag for I2C3
0: no action
1: status flag cleared

Bit 15 **I2C2FC**: clear the illegal access flag for I2C2
0: no action
1: status flag cleared

Bit 14 **I2C1FC**: clear the illegal access flag for I2C1
0: no action
1: status flag cleared

Bit 13 **UART5FC**: clear the illegal access flag for UART5
0: no action
1: status flag cleared

Bit 12 **UART4FC**: clear the illegal access flag for UART4
0: no action
1: status flag cleared

Bit 11 **USART3FC**: clear the illegal access flag for USART3
0: no action
1: status flag cleared

Bit 10 **USART2FC**: clear the illegal access flag for USART2
0: no action
1: status flag cleared

Bit 9 **SPI3FC**: clear the illegal access flag for SPI3
0: no action
1: status flag cleared

Bit 8 **SPI2FC**: clear the illegal access flag for SPI2
0: no action
1: status flag cleared

Bit 7 **IWDGFC**: clear the illegal access flag for IWDG
0: no action
1: status flag cleared

Bit 6 **WWDGFC**: clear the illegal access flag for WWDG
0: no action
1: status flag cleared

Bit 5 **TIM7FC**: clear the illegal access flag for TIM7
0: no action
1: status flag cleared

Bit 4 **TIM6FC**: clear the illegal access flag for TIM6
0: no action
1: status flag cleared

Bit 3 **TIM5FC**: clear the illegal access flag for TIM5
0: no action
1: status flag cleared1: enable

Bit 2 **TIM4FC**: clear the illegal access flag for TIM4
0: no action
1: status flag cleared

Bit 1 **TIM3FC**: clear the illegal access flag for TIM3
0: no action
1: status flag cleared

Bit 0 **TIM2FC**: clear the illegal access flag for TIM2
0: no action
1: status flag cleared

5.7.8 GTZC_TZIC flag clear register 2 (GTZC_TZIC_FCR2)

Address offset: 0x424

Reset value: 0x0000 0000

Secure access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	OTFDEC1FC	EXTIFC	FLASH_REGFC	FLASHFC	RCCFC	DMAMUX1FC	DMA2FC	DMA1FC	SYSCFGFC	PWRFC	RTCFC	OCTOSPI1_REGFC	FMC_REGFC	SDMMC1FC
		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
PKAFC	RNGFC	HASHFC	AESFC	ADCFC	ICACHE_REGFC	TSCFC	CRCFC	DFSDM1FC	SAI2FC	SAI1FC	TIM17FC	TIM16FC	TIM15FC	USART1FC	TIM8FC
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **OTFDEC1FC**: clear the illegal access flag for OTFDEC1
0: no action
1: status flag cleared

Bit 28 **EXTIFC**: clear the illegal access flag for EXTI
0: no action
1: status flag cleared

Bit 27 **FLASH_REGFC**: clear the illegal access flag for FLASH registers
0: no action
1: status flag cleared

Bit 26 **FLASHFC**: clear the illegal access flag for FLASH
0: no action
1: status flag cleared

Bit 25 **RCCFC**: clear the illegal access flag for RCC
0: no action
1: status flag cleared

Bit 24 **DMAMUX1FC**: clear the illegal access flag for DMAMUX1
0: no action
1: status flag cleared

Bit 23 **DMA2FC**: clear the illegal access flag for DMA2
0: no action
1: status flag cleared

Bit 22 **DMA1FC**: clear the illegal access flag for DMA1
0: no action
1: status flag cleared

Bit 21 **SYSCFGFC**: clear the illegal access flag for SYSCFG
0: no action
1: status flag cleared

Bit 20 **PWRFC**: clear the illegal access flag for PWR
0: no action
1: status flag cleared

Bit 19 **RTCFC**: clear the illegal access flag for RTC
0: no action
1: status flag cleared

Bit 18 **OCTOSPI1_REGFC**: clear the illegal access flag for OCTOPSPI1 registers
0: no action
1: status flag cleared

Bit 17 **FMC_REGFC**: clear the illegal access flag for FMC registers
0: no action
1: status flag cleared

Bit 16 **SDMMC1FC**: clear the illegal access flag for SDMMC1
0: no action
1: status flag cleared

Bit 15 **PKAFC**: clear the illegal access flag for PKA
0: no action
1: status flag cleared

Bit 14 **RNGFC**: clear the illegal access flag for RNG
0: no action
1: status flag cleared

Bit 13 **HASHFC**: clear the illegal access flag for HASH
0: no action
1: status flag cleared

Bit 12 **AESFC**: clear the illegal access flag for AES
0: no action
1: status flag cleared

Bit 11 **ADCFC**: clear the illegal access flag for ADC
0: no action
1: status flag cleared

Bit 10 **ICACHE_REGFC**: clear the illegal access flag for ICACHE registers
0: no action
1: status flag cleared

Bit 9 **TSCFC**: clear the illegal access flag for TSC
0: no action
1: status flag cleared

Bit 8 **CRCFC**: clear the illegal access flag for CRC
0: no action
1: status flag cleared

Bit 7 **DFSDM1FC**: clear the illegal access flag for DFSDM1
0: no action
1: status flag cleared

Bit 6 **SAI2FC**: clear the illegal access flag for SAI2
0: no action
1: status flag cleared

Bit 5 **SAI1FC**: clear the illegal access flag for SAI1
0: no action
1: status flag cleared

Bit 4 **TIM17FC**: clear the illegal access flag for TIM17
0: no action
1: status flag cleared

Bit 3 **TIM16FC**: clear the illegal access flag for TIM16
0: no action
1: status flag cleared

Bit 2 **TIM15FC**: clear the illegal access flag for TIM15
0: no action
1: status flag cleared

Bit 1 **USART1FC**: clear the illegal access flag for USART1
0: no action
1: status flag cleared

Bit 0 **TIM8FC**: clear the illegal access flag for TIM8
0: no action
1: status flag cleared

5.7.9 GTZC_TZIC flag clear register 3 (GTZC_TZIC_FCR3)

Address offset: 0x428

Reset value: 0x0000 0000

Secure access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPCBB2_REGFC	SRAM2FC	MPCBB1_REGFC	SRAM1FC	OCTOSPI1_MEMFC	FMC_MEMFC	TZICFC	TZSCFC
								w	w	w	w	w	w	w	w

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

Bit 7 **MPCBB2_REGFC**: clear the illegal access flag for MPCBB2 registers
0: no action
1: status flag cleared

Bit 6 **SRAM2FC**: clear the illegal access flag for SRAM2
0: no action
1: status flag cleared

Bit 5 **MPCBB1_REGFC**: clear the illegal access flag for MPCBB1 registers
0: no action
1: status flag cleared

Bit 4 **SRAM1FC**: clear the illegal access flag for SRAM1
0: no action
1: status flag cleared

Bit 3 **OCTOSPI1_MEMFC**: clear the illegal access flag for OCTOSPI memory interface
0: no action
1: status flag cleared

Bit 2 **FMC_MEMFC**: clear the illegal access flag for FMC NAND and FMC NOR memory interface
0: no action
1: status flag cleared

Bit 1 **TZICFC**: clear the illegal access flag for TZIC
0: no action
1: status flag cleared

Bit 0 **TZSCFC**: clear the illegal access flag for TZSC
0: no action
1: status flag cleared

5.7.10 GTZC_TZIC register map and reset values

Table 29. GTZC_TZIC 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
0x400	GTZC_TZIC_IER1	SPI1IE	TIM1IE	COMPIE	VREFBUFIE	UCPD1IE	USBFSIE	FDCAN1IE	LPTIM3IE	LPTIM2IE	I2C4IE	LPUART1IE	LPTIM1IE	OPAMPIE	DAC1IE	CRSIE	I2C3IE	I2C2IE	I2C1IE	UART5IE	UART4IE	USART3IE	USART2IE	SPI3IE	SPI2IE	IWDGIE	WWDGIE	TIM7IE	TIM6IE	TIM5IE	TIM4IE	TIM3IE	TIM2IE
	Reset value	0	0	0	0	0	0	0	0	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	GTZC_TZIC_IER2	Res.	Res.	OTFDEC1IE	EXTIIE	FLASH_REGIE	FLASHIE	RCCIE	DMAMUX1IE	DMA2IE	DMA1IE	SYSCFGIE	PWRIE	RTCIE	OCTOSPI1_REGIE	FMC_REGIE	SDMMC1IE	PKAIE	RNGIE	HASHIE	AESIE	ADCIE	ICACHE_REGIE	TSCIE	CRCIE	DFSDM1IE	SAI2IE	SAI1IE	TIM17IE	TIM16IE	TIM15IE	USART1IE	TIM8IE	
	Reset value			0	0	0	0	0	0	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 29. GTZC_TZIC 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
0x408	GTZC_TZIC_IER3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	OCTOSPI1_MEMIE	FMC_MEMIE	TZICIE	TZSCIE
	Reset value																									0	0	0	0	0	0	0	0
0x40C	Reserved	Reserved																															
0x410	GTZC_TZIC_SR1	SPI1F	TIM1F	COMPF	VREFBUFF	UCPD1F	USBFSF	FDCAN1F	LPTIM3F	LPTIM2F	I2C4F	LPUART1F	LPTIM1F	OPAMPF	DAC1F	CRSF	I2C3F	I2C2F	I2C1F	UART5F	UART4F	USART3F	USART2F	SPI3F	SPI2F	IWDGF	WWDGF	TIM7F	TIM6F	TIM5F	TIMIF	TIM3F	TIM2F
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x414	GTZC_TZIC_SR2	Res.	Res.	OTFDEC1F	EXTIF	FLASH_REGF	FLASHF	RCCF	DMAMUX1F	DMA2F	DMA1F	SYSCFGF	PWRF	RTCF	OCTOSPI1_REGF	FMC_REGF	SDMMC1F	PKAF	RNGF	HASHF	AESF	ADCF	ICACHE_REGF	TSCF	CRCF	DFSDM1F	SAI2F	SAI1F	TIM17F	TIM16F	TIM15F	USART1F	TIM8F
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	GTZC_TZIC_SR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	OCTOSPI1_MEMF	FMC_MEMF	TZICF	TZSCF
	Reset value																									0	0	0	0	0	0	0	0
0x41C	Reserved	Reserved																															
0x420	GTZC_TZIC_FCR1	SPI1FC	TIM1FC	COMPFC	VREFBUFFC	UCPD1FC	USBFSFC	FDCAN1FC	LPTIM3FC	LPTIM2FC	I2C4FC	LPUART1FC	LPTIM1FC	OPAMPFC	DAC1FC	CRSFC	I2C3FC	I2C2FC	I2C1FC	UART5FC	UART4FC	USART3FC	USART2FC	SPI3FC	SPI2FC	IWDGFC	WWDGFC	TIM7FC	TIM6FC	TIM5FC	TIM4FC	TIM3FC	TIM2FC
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x424	GTZC_TZIC_FCR2	Res.	Res.	OTFDEC1FC	EXTIFC	FLASH_REGFC	FLASHFC	RCCFC	DMAMUX1FC	DMA2FC	DMA1FC	SYSCFGFC	PWRFC	RTCFC	OCTOSPI1_REGFC	FMC_REGFC	SDMMC1FC	PKAFC	RNGFC	HASHFC	AESFC	ADCFC	ICACHE_REGFC	TSCFC	CRCFC	DFSDM1FC	SAI2FC	SAI1FC	TIM17FC	TIM16FC	TIM15FC	USART1FC	TIM8FC
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x428	GTZC_TZIC_FCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPCBB2_REGFC	SRAM2FC	MPCBB1_REGFC	SRAM1FC	OCTOSPI1_MEMFC	FMC_MEMFC	TZICFC	TZSCFC
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3 on page 87* for the register boundary addresses.

6 Embedded Flash memory (FLASH)

6.1 Introduction

The Flash memory interface manages accesses to the Flash memory, maximizing throughput to the CPU, instruction cache and DMAs. It implements the Flash memory erase and program operations as well as the read and write protection mechanisms. It also implements the security and privilege access control features.

6.2 FLASH main features

- Up to 512 Kbytes 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).
- Bank swapping: the address mapping of the user Flash memory of each bank can be swapped.
- Readout protection activated by option (RDP) byte.
- Four write protection areas (two per bank when DBANK=1 and four for full memory when DBANK=0).
- TrustZone support:
  - Two secure areas (1 per bank when DBANK=1, 2 for all memory when DBANK=0)
  - Two secure HDP (hide protection) areas part of the secure areas (one per bank when DBANK=1, two for all memory when DBANK=0).
- Error code correction ECC: 8 bits per 64-bit double-word:
  - DBANK=1: 8 + 64 = 72 bits, two bits detection, one bit correction
  - DBANK=0: (8+64) + (8+64) = 144 bits, two bits detection, one bit correction per 64-bit double-word.
- Option-byte loader.
- Low-power mode.
- Privileged and unprivileged support.

6.3 Flash memory functional description

6.3.1 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).
- Dual-bank mode when DBANK bit is set:
  - 256 Kbytes organized in two banks for main memory.
  - Page size of 2 Kbytes.
  - 72 bits wide data read (64 bits plus 8 ECC bits).
  - Bank and mass erase.
- Single-bank mode when DBANK is reset:
  - 512 Kbytes organized in one single bank for main memory.
  - Page size of 4 Kbytes.
  - 144 bits wide data read (64 bits plus 2x 8 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 two banks of 256 Kbytes, and each bank is organized as follows:
    The main memory block containing 128 pages of 2 Kbytes
  - When dual bank is disabled (DBANK bit reset), the main memory block is organized as one single bank of 512 Kbytes as follows:
    The main memory block containing 128 pages of 4 Kbytes.
- An Information block containing:
  - 32 Kbytes for system memory. The area is reserved for use by STMicroelectronics and contains the bootloader that is used to reprogram the Flash memory through one of the following interfaces: USART1, USART2, USART3, USB (DFU), I2C1, I2C2, I2C3, SPI1, SPI2, SPI3. It is programmed by STMicroelectronics when the device is manufactured, and protected against spurious write/erase operations. For further details, please refer to the application note *STM32 microcontroller system memory boot mode (*AN2606) available from *www.st.com*.
  - 10 Kbytes for root secure services (RSS).
  - 512 bytes 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.
  - 4 Kbytes of option bytes for user configuration. Unlike user Flash memory and system memory, it is not mapped to any memory address and can be accessed only through the Flash register interface.

The memory organization is based on a main area and an information block as shown in *Table 30: Flash module - 512 KB dual bank organization (64 bits read width)* and *Table 31: Flash module - 512 KB single bank organization (128 bits read width)*.

Table 30. Flash module - 512 KB dual bank organization (64 bits read width)

Flash area		Flash memory address	Size	Name
Main memory	Bank 1	0x0800 0000 - 0x0800 07FF	2 Kbytes	Page 0
		0x0800 0800 - 0x0800 0FFF	2 Kbytes	Page 1
		-	-	-
		-	-	-
		-	-	-
		0x0803 F800 - 0x0803 FFFF	2 Kbytes	Page 127
	Bank 2	0x08040000 - 0x0804 07FF	2 Kbytes	Page 0
		0x0804 0800 - 0x0804 0FFF	2 Kbytes	Page 1
		-	-	-
		-	-	-
		-	-	-
		0x0807 F800 - 0x0807 FFFF	2 Kbytes	Page 127
Non-secure Information block(1)		0x0BF9 0000 - 0x0BF9 7FFF	32 Kbytes	System memory
		0x0BFA 0000 - 0x0BFA 01FF	512 bytes	OTP area
Secure Information block		0x0FF8 0000 - 0x0FF8 1FFF	8 Kbytes	RSS
		0x0FF8 2000 - 0x0FF8 27FF	2 Kbytes	RSS library

1. When the TrustZone is enabled (TZEN = 1), the non-secure information block is accessible only with a non-secure access. This means that in order to be able to access any address in this region, it should be configured as non-secure through the SAU. This region is also accessible when booting from RSS.

Note: The secure information block is only available when TrustZone is active.

Table 31. Flash module - 512 KB single bank organization (128 bits read width)

Flash area	Flash memory address	Size	Name
Main memory	0x0800 0000 - 0x0800 0FFF	4 Kbytes	Page 0
	0x0800 1000 - 0x0800 1FFF	4 Kbytes	Page 1
	0x0800 2000 - 0x0800 2FFF	4 Kbytes	Page 2
	-	-	-
	-	-	-
	-	-	-
	0x0807 F000 - 0x0807 FFFF	4 Kbytes	Page 127
Non-Secure Information block(1)	0x0BF9 0000 - 0x0BF9 7FFF	32 Kbytes	System memory
	0x0BFA 0000 - 0x0BFA 01FF	512 bytes	OTP area
Secure Information block	0x0FF9 0000 - 0x0FF9 1FFF	8 Kbytes	RSS
	0x0FF9 2000 - 0x0FF9 27FF	2 Kbytes	RSS library

1. When the TrustZone is enabled (TZEN = 1), the non-secure information block is accessible only with a non-secure access. This means that in order to be able to access any address in this region, it should be configured as non-secure through the SAU. This region is also accessible when booting from RSS.

Note: The secure information block is only available when TrustZone is active.

6.3.2 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 per 64 double words
- 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[17:0] and BK_ECC in the FLASH_ECCR register. ADDR_ECC[18] and 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.

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.

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 two 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 two times double word is saved into ADDR_ECC[18:0] in FLASH_ECCR. ADDR_ECC[18:0] contains an address of a two 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.

6.3.3 Read access latency

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the *FLASH registers* according to the frequency of the CPU clock (HCLK) and the internal voltage range of the device V_{CORE}. Refer to *Section 8.2.5: Dynamic voltage scaling management*. *Table 32* shows the correspondence between wait states and CPU clock frequency.

Table 32. Number of wait states according to CPU clock (HCLK) frequency

Wait states (WS) (Latency)	HCLK (MHz)		
	V_{CORE} Range 0	V_{CORE} Range 1	V_{CORE} Range 2
0 WS (1 CPU cycles)	≤20	≤20	≤8
1 WS (2 CPU cycles)	≤40	≤40	≤16
2 WS (3 CPU cycles)	≤60	≤60	≤26
3 WS (4 CPU cycles)	≤80	≤80	-
4 WS (5 CPU cycles)	≤100	-	-
5 WS (6 CPU cycles)	≤110	-	-

After reset, the CPU clock frequency is 4 MHz and 0 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 *Section 6.9: FLASH registers*.
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. Modify the CPU clock source by writing the SW bits in the RCC_CFGR register.
4. If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.
5. 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. 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.
4. Program the new number of wait states to the LATENCY bits in *Section 6.9: FLASH registers*.
5. Check that the new number of wait states is used to access the Flash memory by reading the FLASH_ACR register.

6.3.4 Low-voltage read

When the external SMPS option is used, the Flash must be programmed in low voltage read mode.

- Voltage scaling range must be in range 2.
- Unlock the LVEN bit by the following procedure:
  - Write LVEKEY1 = 0xF4F5F6F7 in the FLASH_LVEKEYR.
  - Write LVEKEY2 = 0x0A1B2C3D in the FLASH_LVEKEYR.
- Set the LVEN bit in the FLASH_ACR register.
- Check the LVEN bit is set.
- Read back the LVEN bit the FLASH_ACR register

6.3.5 Flash program and erase operations

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

The **in-circuit programming (ICP)** method is used to update the entire contents of the Flash memory, using the JTAG, SWD protocol or the bootloader 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 must have been previously programmed in the Flash memory using ICP.

The contents of the Flash memory are not guaranteed if a device reset occurs during a Flash memory program or erase operation.

In dual bank mode, 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 6.3.9: 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 0 and 1.

Note: At power-on reset or a system reset, the main regulator voltage range 2 is selected by default. Consequently, the voltage scaling range must be programmed to range 0 or range 1 via VOS[1:0] bits in the PWR_CR1 register prior to any Flash erase and programming operation.

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 been completed.

The MCU supports Arm® TrustZone® which defines secure and non-secure areas in Flash. All program and erase operations can be performed in secure mode through the secure registers or in non-secure mode through the non-secure registers. For more information, refer to *Section 6.5: Flash TrustZone security and privilege protections*.

Unlocking the secure/non-secure Flash control register

After reset, write is not allowed in the secure/non-secure Flash control register FLASH_NSCR/FLASH_SECCR 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 secure key register (FLASH_SECKEYR)* or *Flash non-secure key register (FLASH_NSKEYR)*.
2. Write KEY2 = 0xCDEF89AB in the *Flash secure key register (FLASH_SECKEYR)* or *Flash non-secure key register (FLASH_NSKEYR)*.

Any wrong sequence locks up the FLASH_NSCR/FLASH_SECCR 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_NSCR and FLASH_SECCR registers can be locked again by software by setting the NSLOCK and SECLOCK bit in the FLASH_NSCR and FLASH_SECCR register respectively.

*Note: The FLASH_*NSCR and FLASH_SECCR *registers cannot be written when the NSBSY or SECBSY bits are set. Any attempt to write to them with the NSBSY or SECBSY bit causes the AHB bus to stall until the NSBSY and SECBSY bits are cleared.*

Flash secure and non-secure busy flags

The SECBSY and NSBSY flags are both set when a secure or non-secure Flash operation is started:

- Erase operation: setting the NSSTRT in the *Flash non-secure control register (FLASH_NSCR)* or setting the SECSTRT in the *Flash secure control register (FLASH_SECCR)*.
- Write operation: setting the NSPG or SECPG bit in the FLASH_NSCR or FLASH_SECCR register respectively and writing a double-word in the Flash memory.
- Option bytes programming: setting the OPTSTRT in the FLASH_NSCR.

6.3.6 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). The erase operation is either secure or non-secure.

Non-secure page erase

To erase a non-secure page, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the NSBSY bit in the *Flash status register (FLASH_NSSR)*.
2. Check and clear all non-secure error programming flags due to a previous programming. If not, NSPGSERR is set.
3. In dual-bank mode (DBANK option bit is set), set the NSPER bit and select the non-secure page to erase (NSPNB) with the associated bank (NSBKER) in the FLASH_NSCR. In single-bank mode (DBANK option bit is reset), set the NSPER bit and select the page to erase (NSPNB). The NSBKER bit in the FLASH_NSCR must be kept cleared.
4. Set the NSSTRT bit in the FLASH_NSCR register.
5. Wait for the NSBSY bit to be cleared in the FLASH_NSSR register.

Secure page erase

To erase a secure page, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the SECBSY bit in the *Flash status register (FLASH_SECSR)*.
2. Check and clear all secure error programming flags due to a previous programming. If not, SECPGSERR is set.
3. In dual-bank mode (DBANK option bit is set), set the SECPER bit and select the secure page to erase (SECPNB) with the associated bank (SECBKER) in the FLASH_SECCR. In single-bank mode (DBANK option bit is reset), set the SECPER bit and select the page to erase (SECPNB). The SECBKER bit in the FLASH_SECCR must be kept cleared.
4. Set the SECSTRT bit in the FLASH_SECR register.
5. Wait for the SECBSY bit to be cleared in the FLASH_SECSR register.

Note: If the page erase is part of write-protected area (by WRP), NSWRPERR or SECWRPERR is set and the page erase request is aborted.

Non-secure bank 1, bank 2 mass erase (available only in dual-bank mode when DBANK=1)

To perform a non-secure bank mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the NSBSY bit in the FLASH_NSSR register.
2. Check and clear all non-secure error programming flags due to a previous programming. If not, NSPGSERR is set.
3. Set the NSMER1 bit or NSMER2 (depending on the bank) in the FLASH_NSCR register. Both banks can be selected in the same operation, in that case it corresponds to a mass erase.
4. Set the NSSTRT bit in the FLACH_NSCR register.
5. Wait for the NSBSY bit to be cleared in the FLASH_NSSR register.
6. The NSMER1 or NSMER2 bits can be cleared if no more non-secure bank erase is requested.

Secure bank 1, bank 2 mass erase (available only in dual-bank mode when DBANK=1)

To perform a secure bank mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the SECBSY bit in the FLASH_SECSR register.
2. Check and clear all secure error programming flags due to a previous programming. If not, SECPGSERR is set.
3. Set the SECMER1 bit or SECMER2 (depending on the bank) in the FLASH_SECCR register. Both banks can be selected in the same operation, in that case it corresponds to a mass erase.
4. Set the SECSTRT bit in the FLACH_SECCR register.
5. Wait for the SECBSY bit to be cleared in the FLASH_SECSR register
6. The SECMER1 or SECMER2 bit can be cleared if no more secure bank erase is requested.

Non-secure mass erase

To perform a non-secure mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the NSBSY bit in the FLASH_NSSR register.
2. Check and clear all non-secure error programming flags due to a previous programming. If not, NSPGSERR is set.
3. Set the NSMER1 bit and NSMER2 bits in the FLASH_NSCR register.
4. Set the NSSTRT bit in the FLACH_NSCR register.
5. Wait for the NSBSY bit to be cleared in the FLASH_NSSR.
6. The NSMER1 and NSMER2 bit can be cleared if no more non-secure mass erase is requested.

Note: When DBANK=0, if only the NSMERA or the NSMERB bit is set, NSPGSERR 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), NSWRPERR is set and the mass erase request is aborted (for both banks if both are selected).

Secure mass erase

To perform a secure mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the SECBSY bit in the FLASH_SECSR register.
2. Check and clear all error programming flags due to a previous programming. If not, SECPGSERR is set.
3. Set the SECMER1 bit and SECMER2 bits in FLASH_SECCR register.
4. Set the SECTRT bit in the FLACH_SECR register.
5. Wait for the SECBSY bit to be cleared in the FLASH_SECSR.
6. The SECMER1 and SECMER2 bit can be cleared if no more secure mass erase is requested.

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

When DBANK=0, if only the SECMERA or the SECMERB bit is set, SECPGSERR 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), SECWRPERR is set and the mass erase request is aborted (for both banks if both are selected).

6.3.7 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 the NSPROGERR or SECPROGERR flag in the Flash.

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

- Any attempt to write byte or half-word sets NSSIZERR or SECSIZERR flag in the FLASH_NSSR or FLASH_SECSR register.
- Any attempt to write a double word which is not aligned with a double word address sets NSPGAERR or SECPGAERR flag in the FLASH_NSSR or FLASH_SECSR register.

Non-secure programming

The Flash memory programming sequence is as follows:

1. Check that no Flash main memory operation is ongoing by checking the NSBSY bit in the FLASH_NSSR.
2. Check and clear all non-secure error programming flags due to a previous programming. If not, NSPGSERR is set.
3. Set the NSPG bit in the FLASH_NSCR register.
4. Perform the data write operation at the desired memory non-secure address, or in the OTP area. Only double word can be programmed.
   – Write a first word in an address aligned with double word
   – Write the second word in the same double-word.
5. Wait until the NSBSY bit is cleared in the FLASH_NSSR register.
6. Check that NSEOP flag is set in the FLASH_NSSR register (meaning that the programming operation has succeed), and clear it by software.
7. Clear the NSPG bit in the FLASH_NSSR register if there no more programming request anymore.

Secure programming

The Flash memory programming sequence is as follows:

1. Check that no Flash main memory operation is ongoing by checking the SECBSY bit in the FLASH_SECSR.
2. Check and clear all secure error programming flags due to a previous programming. If not, SECPGSERR is set.
3. Set the SECPG bit in the FLASH_SECCR register.
4. Perform the data write operation at the desired memory secure address. Only double word can be programmed.
   – Write a first word in an address aligned with double word
   – Write the second word in the same double-word.
5. Wait until the SECBSY bit is cleared in the FLASH_SECSR register.
6. Check that SECEOP flag is set in the FLASH_SECSR register (meaning that the programming operation has succeed), and clear it by software.
7. Clear the SECPG bit in the FLASH_SECSR register if there is no more programming request anymore.

Note: When the Flash interface has received a good sequence (a double word), programming is automatically launched and SECBSY/NSBSY bit is set. The internal oscillator HSI16 (16 MHz) is enabled automatically when SECPG/NSPG bit is set, and disabled automatically when SECPG/NSPG 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.

6.3.8 Flash errors flags

Flash programming errors

Several kind of errors can be detected during a secure and non-secure operations. In case of error, the Flash operation (programming or erasing) is aborted. The secure errors flags are only set during a secure operation and non-secure flags are only set during a non-secure access.

- **SECPROGERR, NSPROGERR**: secure /non-secure programming error.

  It is set when the double word to program is pointing to an address:
  - Not previously erased
  - Already fully programmed to “0”
  - Already partially programmed (contains “0” and “1”) and the new value to program is not full zero.
  - For OTP programming, the **SECPROGERR, NSPROGERR** flag is set when the address is already partially programmed (contains “0” and “1”).
- **SECSIZERR, NSSIZERR**: secure/non-secure size programming error.

  Only a double word can be programmed and only 32-bit data can be written. SECSIZERR or NSSIZERR flag is set if a byte or a half-word is written.
- **SECPGAERR**, **NSPGAERR**: secure/non-secure alignment programming error.

  It is set when 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.
- **SECPGSERR**: Secure programming sequence error.

  SECPGSERR is set if one of the following conditions occurs during a secure erase or program operation:
  - A data is written when SECPG is cleared.
  - A program operation is requested during erase: SECPG is set while SECMER1 or SECMER2 or SECPER is set.
  - In the erase sequence: SECPG is set while SECSTRT is set
  - If SECSTRT is set with SECMER1 and SECMER2 and SECPER are cleared.
  - If secure page and mass erase are requested at the same time: SECSTRT and SECPER are set and SECMER1 or SECMER2 is set.
  - In single-bank mode (DBANK=0), if SECSTRT is set and only SECMER1 or SECMER2 is set.
  - If SECPROGERR, SECSIZERR, SECPGAERR, SECWRPERR or SECPGSERR is already set due to a previous programming error.

- **NSPGSERR**: non-secure programming sequence error.

  NSPGSERR is set if one of the following conditions occurs during a non-secure erase or program operation:
  - A data is written when NSPG is cleared.
  - A program operation is during erase: NSPG is set while NSMER1 or NSMER2 or NSPER is set.
  - In the erase sequence: NSPG is set while NSSTRT is set.
  - If NSSTRT is set with NSMER1 and NSMER2 and NSPER are cleared.
  - If non-secure page and mass erase are requested at the same time: NSSTRT and NSPER are set and NSMER1 or NSMER2 is set.
  - In single-bank mode (DBANK=0), if NSSTRT is set and only NSMER1 or NSMER2 is set.
  - If NSPROGERR, NSSIZERR, NSPGAERR, NSWRPERR, NSPGSERR or OPTWERR is already set due to a previous programming error.
  - If NSSTRT is set by a secure access.
  - If NSSTRT and OPTSTRT are set at the same time.
- **SECWRPERR**: secure write protection error.

  SEWRPERR is set if one of the following conditions occurs:
  - A secure program or erase on a non-secure page or a write protected area (WRP).
  - A secure bank erase or mass erase when one page or more is protected by WRP or HDP area with access disabled.
  - Refer to *Table 47* to *Table 49* for all the conditions of SECWRPERR flag setting.
- **NSWRPERR**: non-secure write protection error.

  NSWRPERR is set if one of the following conditions occurs:
  - A non-secure program or erase in a non-secure write protected area (WRP) or in a secure area (Secure watermark-based, HDP, Secure block-based).
  - A non-secure bank erase or mass erase when one page or more is protected by WRP or a secure area (SECWM, HDP, SECBB).
  - Refer to *Table 47* to *Table 49* for all the conditions of NSWRPERR flag setting.
- **OPTWERR**: option bytes write error.

  OPTWERR is set if when user option bytes are modified with an invalid configuration. It set when attempt:
  - To program an invalid secure watermark-based area. Refer to *Table 35: Secure watermark-based area*
  - To set or unset TZEN option bit without being in correct RDP level (refer to *Rules for modifying specific option bytes*).
  - To modify DBANK option bit while the Flash is secure (watermak or block-based)
  - To set SWAP_BANK option bit while DBANK is cleared.
  - To modify SWAP_BANK option bit while BOOT_LOCK and TZEN are set.
  - To modify SECBOOTADD0 option bit while BOOT_LOCK is set.
  - To modify DB256 option bit while BOOT_LOCK and TZEN are set.
  - Attempt to modify the option bytes, except the SWAP_BANK option bit ,when the readout protection (RDP) is set to 2.

If an error occurs during a a secure or non-secure program or erase operation, one of the following programming error flags is set:

- Non-secure programming error flags: NSPROGERR, NSSIZERR, NSPGAERR, NSPGSERR, OPTWRERR or NSWRPERR is set in the FLASH_NSSR register.
  - If the non-secure error interrupt enable bit NSERRIE is set in the *Flash non-secure control register (FLASH_NSCR)*, an interrupt is generated and the operation error flag NSOPERR is set in the FLASH_NSSR register.
- Secure programming error flags: SECPROGERR, SECSIZERR, SECPGAERR, SECPGSERR or SECWRPERR is set in the FLASH_SECSR register.
  - If the secure error interrupt enable bit SECERRIE is set in the *Flash secure control register (FLASH_SECCR)*, an interrupt is generated and the operation error flag SECOPERR is set in the FLASH_SECSR 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.

6.3.9 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 reset, allowing read-while-write operations. This feature permits to perform a read operation from one bank while 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 NSBSY or SECBSY bit in the FLASH_NSSR or FLASH_SECSR register (NSBSY, SECBSY are set when erase/program operation is on going in bank 1 or bank 2).
2. Set NSPER or SECPER bit, NSPSB or SECPSB to select the non-secure or secure page and NSBKER or SECBER to select the bank following the security state non-secure or secure.
3. Set the NSSTRT or SECSTRT bit in the FLASH_NSCR/FLASH_SECCR register.
4. Wait for the NSBSY or SECBSY bit to be cleared (or use the NSEOP or SECEOP 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 NSBSY/SECBSY bit in the FLASH_NSSR/FLASH_SECSR register (NSBSY, SECBSY are active when erase/program operation is on going in bank 1 or bank 2).
2. Non-secure bank erase, set the NSMER1 or NSMER2 in the FLASH_NSCR register. For secure bank erase, set the SECMER1 or SECMER2 in the FLASH_SECCR register.
3. Set the NSSTRT/SECSTRT bit in the FLASH_NSCR/FLASH_SECCR register.
4. Wait for the NSBSY or SECBSY bit to be cleared (or use the NSEOP or SECEOP 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 NSBSY/SECBSY bit in the FLASH_NSSR/FLASH_SECSR register (NSBSY, SECBSY are active when erase/program operation is on going in bank 1 or bank 2).
2. Set the NSPG or SECPG bit in the FLASH_NSCR/FLASHSECCR register.
3. Perform the data write operations at the desired address memory inside the main memory block or OTP area.
4. Wait for the NSBSY or SECBSY bit to be cleared (or use the NSEOP or SECEOP interrupt).

Note: Due to Cortex M33 unified C-Bus, user software must ensure to not stall C-Bus with multiple consecutive writes. It is recommended to wait the NSBSY/SECBSY flag to be cleared before programming the next double word.

6.4 Flash memory option bytes

6.4.1 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 6.4.2: Option bytes programming*).The user option bytes are accessible through the Flash interface registers interface.

Table 33 describes the organization of all user option bytes available in the Flash interface registers.

Table 33. User option byte organization mapping

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	Register map
TZEN	Res.	Res.	PA15_PUPEN	nBOOT0	nSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	DBANK	DB256K	SWAP_BANK	WWDG_SW	IWDG_STDBY	IWDG_STOP	IDWG_SW	Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	Res.	BOR_LEV[2:0]	RDP										*Section 6.9.12: Flash option register (FLASH_OPTR)*
NSBOOTADD0[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.	*Section 6.9.13: Flash non-secure boot address 0 register (FLASH_NSBOOTADD0R)*
NSBOOTADD1[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.	*Section 6.9.14: Flash non-secure boot address 1 register (FLASH_NSBOOTADD1R)*
SECBOOTADD0[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK	*Section 6.9.15: Flash secure boot address 0 register (FLASH_SECBOOTADD0R)*
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_PSTRT[6:0]							*Section 6.9.16: Flash bank 1 secure watermak1 register (FLASH_SECWM1R1)*
HDP1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	*Section 6.9.17: Flash secure watermak1 register 2 (FLASH_SECWM1R2)*
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_PSTRT[6:0]							*Section 6.9.18: Flash WPR1 area A address register (FLASH_WRP1AR)*
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_PSTRT[6:0]							*Section 6.9.19: Flash WPR1 area B address register (FLASH_WRP1BR)*
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_PSTRT[6:0]							*Section 6.9.20: Flash secure watermak2 register (FLASH_SECWM2R1)*
HDP2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	*Section 6.9.21: Flash secure watermak2 register 2 (FLASH_SECWM2R2)*

Table 33. User option byte organization mapping (continued)

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	Register map
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_PSTRT[6:0]							*Section 6.9.22: Flash WPR2 area A address register (FLASH_WRP2AR)*
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_PSTRT[6:0]							*Section 6.9.23: Flash WPR2 area B address register (FLASH_WRP2BR)*

6.4.2 Option bytes programming

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

1. Unlock the FLASH_NSCR with the NSLOCK clearing sequence (refer to *Unlocking the secure/non-secure Flash control register*).
2. Write OPTKEY1 = 0x08192A3B in the FLASH_OPTKEYR register.
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 NSLOCK is set by software, OPTLOCK is automatically set too.

Option bytes modification sequence

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

1. Check that no Flash memory operation is on going by checking the NSBSY bit in the FLASH_NSSR register.
2. Clear OPTLOCK option lock bit with the clearing sequence described above.
3. Write the desired options value in the options registers.
4. Set the options start bit OPTSTRT in the *Flash non-secure control register (FLASH_NSCR)*.
5. Wait for the NSBSY bit to be cleared.
6. Set the OBL_LAUNCH option bit to start option bytes loading.

Note: If the OPTWERR or NSPGSERR error bit is set, the old option byte values are kept.

Option byte loading

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

- When OBL_LAUNCH bit is set in the *Flash non-secure control register (FLASH_NSCR)*.
- After a power reset (BOR reset or exit from Standby/Shutdown modes).

On system reset rising, internal option registers are copied into option registers. These registers are also used to modify the option bytes. If these registers are not modified by

user, they reflects the options states of the system. See *Section 6.4.2: Option bytes programming* 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.

- If any secure Flash protection is enabled (watemark or block-based), all must be disabled.
- Disable the instruction cache if it is enabled
- 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=0, perform a mass erase.
  - Start a new programing of code in 64 bits mode with DBANK=1 memory mapping.
  - Set the new secure protection 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.

- If any secure Flash protection is enabled (watemark or block-based), all must be disabled.
- Disable the instruction cache if it is enabled
- 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 secure protection if needed.

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

Rules for modifying specific option bytes

Some of the option byte field must respect specific rules before being updated with new values. These option bytes, as well as the associated constraints, are described below:

- TZEN option bit
  - TZEN can only be set on RDP level 0.
  - Deactivation of TZEN must be done at the same time as RDP regression to level 0 (from level 1 to level 0 or from level 0.5 to level 0).
- BOOT_LOCK option bit
  - BOOT_LOCK can be set without any constraint.
  - It is not possible to deactivate the BOOT_LOCK option bit.
- SWAP_BANK option bit
  - It is not possible to set the SWAP_BANK in single-bank mode (DBANK=0).
  - It can not be modified when BOOT_LOCK and TZEN option bit are set.
- SECBOOTADD0 option bytes
  - It can not be modified when BOOT_LOCK option bit is set.
- DB256K option bit
  - It is not possible to modify the DB256K when BOOT_LOCK and TZEN option bits are set.
  - In single-bank mode (DBANK=0).
- DBANK option bit
  - It can only be modified when all secure protections are disabled (secure watermark or block-based area).
- SECWMx_PSTRT[6:0], SECWMxPEND[6:0], HDPx_PEND[6:0], HDPxEN option bytes
  - It can only be modified when HDPxACCDIS bit is cleared. When it is set, options bytes listed in the table below are locked and can not be modified until next system reset.If the user options modification try to modify one of the those option bytes while HDPxACCDIS bit is set, the option bytes modification is discarded without error flag.

If the above user options modification tries to set or modify one of the listed option bytes without following their associated rules, the option bytes modification is discarded and the OPTWERR error flag is set.

Rules of RDP level regression

When TrustZone is active, in case of RDP level 0.5 or level 1, the RDP regression to level 0 or level 0.5 can only be done by the debug interface or by the system bootloader.

6.5 Flash TrustZone security and privilege protections

6.5.1 TrustZone security protection

The global TrustZone system security is activated by setting the TZEN option bit in the FLASH_OPTR register.

When the TrustZone is active (TZEN=1), additional security features are available:

- Secure watermark-based user options bytes defining secure, HDP areas.
- Secure or non-secure block-based areas can be configured on-the-fly after reset. This is a volatile secure area.
- An additional RDP protection: RDP level 0.5.
- Erase or program operation can be performed in secure or non-secure mode with associated configuration bit.

When the TrustZone is disabled (TZEN=0), the above features are deactivated and all secure registers are RAZ/WI.

All other option bytes not listed above, can be modified without any constraints.

Activating TrustZone security

On first TrustZone activation (TZEN is modified from 0 to 1), the secure watermark-based user options bytes are set to default secure state: All Flash is secure, no HDP area. Refer to *Table 34: Default secure option bytes after TZEN activation*.

Table 34. Default secure option bytes after TZEN activation

DBANK option bit	Secure watermark option bytes values after OBL when TZEN is activated (from 0 to 1)	Security attribute
0	SECWM1_PSTRT = 0 SECWM1_PEND = 0x7F SECWM2_PSTRT = 0x7F SECWM2_PEND = 0	All Flash memory is secure
1	SECWMx_PSTRT = 0 SECWMx_PEND = 0x7F	Bank 1 is fully secure Bank 2 is fully secure
0/1	HDPxEN = 0 HDPx_PEND = 0	No secure HDP area

Deactivating TrustZone security

Deactivation of TZEN (from 1 to 0) is only possible when the RDP is changing to level 0 (from level 1 to level 0 or from level 0.5 to level 0).

When the TrustZone is deactivated (TZEN is modified from 1 to 0) after option bytes loading, the following security features are deactivated:

- Watermark-based secure area
- Block-based secure area
- RDP level 0.5
- Secure interrupt
- All secure registers are RAZ/WI.

6.5.2 Secure watermark-based area protection

When TrustZone security is active (TZEN=1), a part of the Flash memory can be protected against non-secure read and write access. Up to two different non-volatile secure areas can be defined by option bytes and can be read or written by a secure access only:

- In single-bank mode, two areas can be selected with a page granularity.
- In dual-bank mode, one area per bank can be selected with a page granularity.

The secure areas are defined by a start page offset and end page offset using the SECWMx_PSTRT and SECWMx_PEND (x=1,2 for area 1 and area 2) option bytes.These offsets are defined in the Secure watermark registers address registers *Flash bank 1 secure watermak1 register (FLASH_SECWM1R1)*, *Flash secure watermak2 register (FLASH_SECWM2R1)*.

The SECWMx_PSTRT and SECWMx_PEND option bytes can only be modified by secure firmware when the HDPxACCDIS bit is reset.

If the HDPxACCDIS bit is set, the SECWMx_PSTRT and SECWMx_PEND cannot be modified until next system reset.

Table 35. Secure watermark-based area

DBANK option bit	Secure watermark option bytes values (x = 1,2)	Secure watermark protection area
0/1	SECWMx_PSTRT > SECWMx_PEND	No secure area
0/1	SECWMx_PSTRT = SECWMx_PEND	One page defined by SECWMx_PSTRT is secure water-mark-based protected
0/1	SECWMx_PSTRT < SECWMx_PEND	The area between SECWMx_P-STRT and SECWMx_PEND is secure watermark-based protected

Caution: Switching a Flash memory area from secure to non-secure does not erase its content. The user secure software must perform the needed operation to erase the secure area before switching an area to non-secure attribute whenever is needed. It is also recommended to flush the instruction cache.

6.5.3 Secure hide protection (HDP)

The secure HDP area is part of the Flash watermark-based secure area. Access to the hide protection area can be denied by setting the HDPxACCDIS bit in FLASH_SECHDPCR register.

When the HDPxACCDIS bit is set, data read, write and instruction fetch on this hide protection area are denied. For example, software code in the secure Flash hide protected area can be executed only once and deny any further access to this area until next system reset. The HDPxACCDIS bit can be only cleared by a system reset.

Up to two non-volatile secure hide protection (HDP) areas can be defined depending of the DBANK mode:

- In single-bank mode, two HDP areas can be selected with a page granularity.
- In dual-bank mode, one HDP area per bank can be selected with a page granularity

The secure HDP area is enabled by the HDPxEN (x=1,2 for area 1 and area 2). When the HDPxEN bit is reset, there is no HDP area. The HDPxEN bit can be set or reset by the secure firmware if the HDPx_ACCDIS bit is reset.

The secure HDP area size is defined by the end page offset using the HDPx_PEND option bytes while the start page offset is already defined by SECWMx_PSTRT option bytes.These offsets are defined in the Secure watermark registers address registers *Flash bank 1 secure watermak1 register (FLASH_SECWM1R1)*, *Flash secure watermak1 register 2 (FLASH_SECWM1R2)*, *Flash secure watermak2 register (FLASH_SECWM2R1)*, *Flash secure watermak2 register 2 (FLASH_SECWM2R2)*.

The HDPxEN and HDPx_PEND option bytes can only be modified by secure firmware when the HDPxACCDIS bit is reset.

If the HDPxACCDIS bit is set, the HDPxEN and HDPx_PEND cannot be modified until next system reset.

If an invalid secure HDP area is defined as described in *Table 36: Secure, HDP protections summary*, the OPTWERR flag error is set and option bytes modification is discarded.

Table 36. Secure, HDP protections summary

Secure, HDP, watermark option bytes values (x = 1,2)		Protections area
HDPxEN bit	**Option bytes**	
x	SECWMx_PSTRT > SECWMx_PEND	No secure area.
0	SECWMx_PSTRT <= SECWMx_PEND	No secure HDP area. Secure area bewteen SECWMx_STRT and SECWMx_PEND
1	SECWMx_STRT <= HDPx-_PEND <= SECWMx_PEND	The area between SECWMx_STRT and HDPx_PEND is secure HDP protected. – If SECWMx_STRT=HDPx_PEND, one page defined in HDPx_PEND is secure HDP protected.
	Others	Invalid secure area. HDP area is defined outside the secure area.

6.5.4 Secure block-based area (SECBB) protection

Any page, non-secure through secure Flash memory watermark option bytes, can be programmed on-the-fly as secure using the block-based configuration registers.

With SECBB, it is not possible to unsecure a secure page through secure watermark option bytes.

In dual-bank mode (DBANK=1):

- FLASH_SECBB1Rx registers are used to configure the security attribute for pages in bank1
- FLASH_SECBB2Rx registers are used to configure the security attribute for pages in bank2

In single-bank mode (DBANK=0):

- the FLASH_SECBB1Rx registers are used to configure the security attribute for pages in all Flash memory.

It is possible to temporary secure a non-secure watermark page by setting corresponding SECBB bit to 1. Setting SECBB bit to 1 on an already secure watermark page has no effect.

To modify a page block-based security attribution, it is recommended to:

- Check that no Flash operation is ongoing on the related page.
- Add ISB instruction after modifying the page security attribute SECBB1/2[i].

6.5.5 Forcing boot from a secure memory address

When TrustZone is enabled by setting the TZEN option bit, the boot space must be in secure area. The SECBOOTADD0[24:0] option bytes are used to select the boot secure memory address. To increase the security and establish a chain of trust, a unique boot entry option can be selected regardless the other boot options. This is done by setting the BOOT_LOCK option bit in the FLASH_SECBOOTADD0R register.

This bit can be set only by a secure access.

Caution: Once set, the BOOT_LOCK option bit cannot be cleared.

6.5.6 Flash security attribute state

The Flash is secure when at least one secure area is defined either by watermark-based option bytes or block-based security registers.

It is possible to override the Flash security state using the SECINV bit in the FLASH_SECCR register.

Table 37. Flash security state

Secure Area	SECINV bit	Flash security state
None	0	Non secure
	1	Secure
Yes	0	Secure
	1	Non secure

- A non-secure access to a a secure Flash memory area is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the FLASHIE illegal access interrupt is enabled in the GTZC_TZIC_IER2 register.
- A non-secure access to a secure Flash register generates an illegal access event. An illegal access interrupt is generated if the FLASH_REGIE illegal access interrupt is enabled in the GTZC_TZIC_IER2 register.

6.5.7 Flash registers privileged and unprivileged modes

The Flash registers can be read and written by privileged and unprivileged accesses depending on PRIV bit in FLASH_PRIVCFGR register.

- When the PRIV bit is reset, all Flash registers could be read and written by both privileged or unprivileged access.
- When the PRIV bit is set, all Flash registers could be read and written by privileged access only. Unprivileged access to a privileged registers is RAZ/WI.

6.6 Secure system memory

6.6.1 Introduction

Secure system memory stores RSS (root secure services) firmware that is programmed by ST during STM32L552xx and STM32L562xx production. The RSS provides secure services to the bootloader and the user firmware. These services are described hereafter in this section.

The RSS services are only available after the user enables the microcontroller TrustZone® feature thanks to TZEN bitfield set to 1 within FLASH_OPTR option byte register.

At boot time, the RSS firmware enables and jumps to bootloader; the RSS provides services to secure user firmware at runtime.

6.6.2 RSS allocates resource to bootloader

When the microcontroller is configured in TrustZone® enabled (option byte register FLASH_OPTR bitfield TZEN set to 1), then the microcontroller must boot on a secure address after reset. According to boot configuration, the boot can be done either from a secure address programmed through SECBOOT0ADDR bitfield of FLASH_SECBOOTADD0R option byte register or from RSS. In this last case, RSS firmware is the first firmware running after boot.

The RSS is then responsible for the microcontroller bootloader resource allocation. The RSS allocates SRAM, system Flash memory, peripherals (USART, I2C, SPI, …), the respective Os and IRQs to non-secure. The bootloader uses these resources to enable communication ports as described in AN2606 – *STM32 microcontroller system memory boot mode*.

Once resource allocation is done, the RSS triggers a secure to non-secure transition jumping to the bootloader, that in turn enables all its communication ports. At this step, the user can connect to the microcontroller either using bootloader communication ports or debug ports (JTAG or SWD).

As a non-secure firmware, the bootloader never accesses to secure resources. However, in RDP level 0 only, the bootloader can access the secure option bytes.

For detailed boot modes, please refer to *Section 3: Boot configuration*.

Table 38 sums up user accesses via bootloader communications ports or JTAG according to RDP level value.

Table 38. User accesses via bootloader or JTAG

RDP level	User access description	User access via	
		Bootloader	JTAG
0	RWX access to non-secure user Flash memory.	Yes	
	RWX access to secure user Flash memory.	No	Yes (secure debug only)
	RW access to Flash memory secure option bytes	Yes	Yes
	RWX access SRAM1	Yes	
	RWX access SRAM2	No	Yes (secure debug only)
0.5	RWX access to non-secure user Flash memory.	Yes	
	RWX access to secure user Flash memory.	No	
	RW access to Flash memory secure option bytes	No	
	RWX access SRAM1	Yes	
	RWX access SRAM2	No	
1	RWX access to non-secure user Flash memory.	No	
	RWX access to secure user Flash memory.	No	
	RW access to Flash memory secure option bytes	No	
	RWX access SRAM1	No	Yes
	RWX access SRAM2	No	
2	RWX access to non-secure user Flash memory.	No (RSS does not jump to Bootloader)	
	RWX access to secure user Flash memory.		
	RW access to Flash memory secure option bytes		
	RWX access SRAM1		
	RWX access SRAM2		

Non secure peripherals, IRQn and IOs allocated to non-secure for bootloader execution are described within *Table 39*.

Table 39. Non-secure peripherals, IRQn and IOs for bootloader execution

HW resource type	Resource description
Peripherals	USART1, USART2, USART3 SPI1, SPI2, SPI3 I2C1, I2C2, I2C3, USBFS FDCAN1 ICACHE IWDG
IRQn	DMA1_Channel3_IRQn, DMA1_Channel5_IRQn, DMA2_Channel2_IRQn, USB_FS_IRQn
IO	USART1 : PA10, PA9 USART2 : PA3, PA2 USART3 : PC11, PC10 SPI1 : PA4, PA5, PA6, PA7 SPI2 : PB12, PB13, PB14, PB15 SPI3 : PB5, PG9, PG10, PG12 I2C1: PB6, PB7 I2C2: PB10, PB11 I2C3: PC0, PC1 USB: PA11, PA12 PDCAN: PB8, PB9

For IRQn and IO bootloader detailed usage, please refer to AN2606 – *STM32 microcontroller system memory boot mode*.

6.6.3 RSSLIB functions

The RSS provides runtime services thanks to RSS library. As other microcontroller peripherals features and mapping, the RSS library functions are exposed to user within the CMSIS device header file provided by the STM32CubeL5 firmware package. Please refer to UM2656 to get more details regarding STM32CubeL5 firmware package. RSS library functions are named RSSLIB functions hereafter.

The user firmware calls RSSLIB functions using RSSLIB_PFUNC C defined macro, that points to a location within non-secure system memory. Hence prior calling RSSLIB functions, the secure user firmware must define a non-secure region above this location within SAU of the Cortex®-M33. This non-secure region starts from RSSLIB_SYS_FLASH_NS_PFUNC_START up to RSSLIB_SYS_FLASH_NS_PFUNC_END. These last addresses are provided within the CMSIS device header file. The user can set this non-secure region either by using the CMSIS system partition header file or by implementing its own code for SAU setup. The CMSIS system partition header file is part of the STM32CubeL5 firmware package.

Note: Some RSSLIB functions are tied to bootloader version (bootloader ID); before calling any RSSLIB function, the user must check within the dedicated section in this document, if it depends or not on a bootloader ID. If the RSSLIB function is dependent on bootloader ID,

the user must read this ID using BL_ID C defined macro from the CMSIS device header file (BL_ID is one-byte long value). Then, the user firmware must call the right function according to bootloader ID value.

RSSLIB functions are split between non-secure callable and secure callable function.

The RSS library functions are described within sections hereafter.

CloseExitHDP_BL90

Bootloader ID:

CloseExitHDP_BL90 function is compliant for bootloader ID 0x90.

Secure attribute:

Secure callable function.

Prototype:

```
uint32_t CloseExitHDP_BL90(uint32_t HdpArea, uint32_t
VectorTableAddr)
```

Arguments:

- HdpArea:
  Input parameter, bitfield that identifies which HDP area to close. Values can be either: RSSLIB_HDP_AREA1_Msk, RSSLIB_HDP_AREA2_Msk or RSSLIB_HDP_AREA1_Msk |RSSLIB_HDP_AREA2_Msk.
- VectorTableAddr:
  Input parameter,address of the next vector table to apply.
  The vector table format is the one used by the Cortex®-M33 core.

Description:

User calls CloseExitHDP_BL90 to close Flash HDP secure memory area and jump to the reset handler embedded within the vector table which address is passed as input parameter.

CloseExitHDP_BL90 sets the SP provided by the passed vector table, however it is up to the caller to first set the new vector table. Then it clears all general-purpose Cortex®-M33 registers (r0, r1, …) before jumping to new vector table reset handler.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), this function returns RSSLIB_ERROR.

Please refer to section *Section 6.5.3: Secure hide protection (HDP)* to get more details on Flash memory HDP protection.

CloseExitHDP_BL91

Bootloader ID:

CloseExitHDP_BL91 function is compliant for bootloader ID 0x91 up to 0x9F.

Secure attribute:

Secure callable function.

Prototype:

```
uint32_t CloseExitHDP_BL91(uint32_t HdpArea, uint32_t
VectorTableAddr)
```

Arguments:

- HdpArea:
  Input parameter, bitfield that identifies which HDP area to close. Values can be either: RSSLIB_HDP_AREA1_Msk, RSSLIB_HDP_AREA2_Msk or RSSLIB_HDP_AREA1_Msk |RSSLIB_HDP_AREA2_Msk.
- VectorTableAddr:
  Input parameter,address of the next vector table to apply.
  The vector table format is the one used by the Cortex®-M33 core.

Description:

The user calls CloseExitHDP_BL91 to close Flash HDP secure memory area and jump to the reset handler embedded within the vector table which address is passed as input parameter.

CloseExitHDP_BL91 sets the SP provided by the passed vector table, however it is up to the caller to first set the new vector table. Then it clears all general-purpose ACortex®-M33 registers (r0, r1, …) before jumping to new vector table reset handler.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), this function returns RSSLIB_ERROR.

Please refer to section *Section 6.5.3: Secure hide protection (HDP)* to get more details on Flash memory HDP protection.

6.7 FLASH memory protection

The Flash interface implements different protection mechanisms:

- Write protection (WRP)
- Readout protection (RDP)
- Secure protection when TrustZone is active
  - Up to two secure watermak-based non-volatile areas
  - Up to two secure block-based volatile areas
  - Up to two secure hide protection areas

6.7.1 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 Kbytes) granularity.
- In dual-bank mode (DBANK=1): two write-protected (WRP) areas can be defined in each bank, with page (2 Kbytes) 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 WPR1 area A address register (FLASH_WRP1AR)*, *Flash WPR1 area B address register (FLASH_WRP1BR)*, *Flash WPR2 area A address register (FLASH_WRP2AR)*, *Flash WPR2 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 0x800] (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 = 0xC5.
  - WRP1A_END = 0xE0.

  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 = 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 secure or non-secure write protection error flag (NSWRPERR or SECWRPERR) is set in the FLASH_NSSR or FLASH_SECSR register. This flag is also set for any write access to:

- System Flash memory.
- OTP area.

Note: When the memory readout protection level is selected (RDP level = 1), it is not possible to program or erase Flash memory (secure or non-secure) 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 40. 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

6.7.2 Readout protection (RDP)

The readout protection is activated by setting the RDP option byte and then, by applying OBL launch or power-on reset to reload the new RDP option byte. The readout protection protects the Flash main memory, the option bytes, the backup registers and the SRAMs.

Readout protection levels when Trustzone is disabled

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

The Flash memory is protected according to the RDP option byte value shown in the table below.

Table 41. Flash memory readout protection status (TZEN=0)

RDP byte value	Readout protection level
0xAA	Level 0
Any value except 0xAA or 0xCC	Level 1
0xCC	Level 2

Level 0: no protection

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

Level 1: readout protection

- **User mode:** code executing in user mode (**boot Flash**) can access Flash main memory, option bytes, SRAMs and backup registers with all operations (read, erase, program).
- **Debug, boot RAM and bootloader modes**: in debug mode or when code is running from boot RAM or bootloader, the Flash main memory, the backup registers and the SRAM2 are totally inaccessible. In Debug and boot RAM modes an intrusion is detected and a read or write access to the Flash or backup SRAM generates a bus error and a hard fault interrupt. The on-the-fly decryption region (OTFDEC on OCTOSPI) is read as zero.

Level 2: no debug

- The protection level 1 is guaranteed.
- All debug features are disabled.
- The boot from SRAM (boot RAM mode) and the boot from system memory (bootloader mode) are no longer available.
- When booting from Flash, all operations are allowed on the Flash main memory. Read, erase and program accesses to Flash memory and SRAMs from user code are allowed.
- Option bytes cannot be programmed nor erased except the SWAP_BANK option bit. Thus, the level 2 cannot be removed: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag OPTWERR is set in the FLASH_NSSR register and an interrupt can be generated.

Note: The debug feature is also disabled under reset.

Table 42. Access status versus protection level and execution modes when TZEN=0

Area	RDP level	User execution (boot from Flash)			Debug/ bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No(2)
	2	Yes	Yes	Yes	NA	NA	NA
System memory(3)	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	NA	NA	NA
Option bytes(4)	1	Yes	Yes(2)	Yes	Yes	Yes(2)	Yes
	2	Yes	No(5)	No	NA	NA	NA
OTP	1	Yes	Yes(6)	NA	Yes	Yes(6)	NA
	2	Yes	Yes(6)	NA	NA	NA	NA
Backup registers	1	Yes	Yes	NA	No	No	NA(7)
	2	Yes	Yes	NA	NA	NA	NA
SRAM2	1	Yes	Yes	NA	No	No	NA(8)
	2	Yes	Yes	NA	NA	NA	NA
OTFDEC regions (OCTOSPI)	1	Yes	Yes	Yes	No(9)	Yes	Yes
	2	Yes	Yes	Yes	NA	NA	NA

1. When the protection level 2 is active, the debug port and the bootloader mode are disabled.
2. The Flash main memory is erased when the RDP option byte regresses from level 1 to level 0.
3. The system memory is only read-accessible whatever the protection level (0, 1 or 2) and execution mode.
4. Option bytes are only accessible through the Flash registers interface and OPSTRT bit.
5. SWAP_BANK option, bit can be modified.
6. OTP can only be written once.
7. The backup registers are erased when RDP changes from level 1 to level 0.
8. All SRAMs are erased when RDP changes from level 1 to level 0.
9. The OTFDEC keys are erased when the RDP option byte changes from level 1 to level 0.

Readout protection levels when Trustzone is enabled

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

The Flash memory is protected according to the RDP option byte value shown in the table below.

Table 43. Flash memory readout protection status (TZEN=1)

RDP byte value	Readout protection level
0xAA	Level 0
0x55	Level 0.5
Any value except 0xAA or 0x55 or 0xCC	Level 1
0xCC	Level 2

Level 0: no protection

Read, program and erase operations into the Flash main memory area are possible. The option bytes, the SRAMs and the backup registers are also accessible by all operations. When booting from RSS, the debug access is disabled while executing RSS code.

Level 0.5: non-secure debug only

All read and write operations (if no write protection is set) from/to the non-secure Flash memory are possible. The debug access to secure area is prohibited. Debug access to non-secure area remains possible.

- **User mode**: code executing in user mode (**boot Flash**) can access Flash main memory, option bytes, SRAMs and backup registers with all operations (read, erase, program).
- **Non-secure debug mode**: non-secure debug is possible when the CPU is in non-secure state. The secure Flash memory, the secure backup registers and SRAMs area are inaccessible; the non-secure Flash memory, the non-secure backup registers and the non-secure SRAMs area remain accessible for debug purpose.
- **RSS mode**: when booting from RSS, the debug access is disabled while executing RSS code.
- **Boot RAM mode**: boot from SRAM is not possible.

Level 1: readout protection

- **User mode:** code executing in user mode (**boot Flash**) can access Flash main memory, option bytes, SRAMs and backup registers with all operations (read, erase, program).
- **Non-secure debug mode**: non-secure debug is possible when the CPU is in non-secure state. However, an intrusion is detected in case of debug access: the Flash main memory, the backup registers and the SRAM2 are totally inaccessible; any read or write access to these memories generates a bus error and a hard fault interrupt. The on-the-fly decryption region (OTFDEC on OCTOSPI) is read as zero.
- **RSS mode**: when booting from RSS, the debug access is disabled while executing the RSS code.
- **Boot RAM mode**: boot from SRAM is no longer possible.

Level 2: no debug

When the readout protection level 2 is set:

- The protection level 1 is guaranteed.
- All debug features are disabled.
- The boot from SRAM (boot RAM mode) and the boot from system memory (bootloader mode) are no longer available.
- Boot from RSS is possible.
- When booting from Flash or RSS, all operations are allowed on the Flash main memory. Read, erase and program accesses to Flash memory and SRAMs from user code are allowed.
- Option bytes cannot be programmed nor erased except the SWAP_BANK option bit. Thus, the level 2 cannot be removed: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag OPTWERR is set in the FLASH_NSSR register and an interrupt can be generated.

Note: The debug feature is also disabled under reset.

Table 44. Access status versus protection level and execution modes when TZEN=1

Area	RDP level	User execution (boot from Flash)			Debug/ bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	0.5	Yes	Yes	Yes	Yes(2)	Yes(2)	Yes(2)
	1	Yes	Yes	Yes	No	No	No(3)
	2	Yes	Yes	Yes	NA	NA	NA
System memory(4)	0.5	Yes	No	No	Yes	No	No
	1	Yes	No	No	Yes	No	No
	2	Yes	Yes	Yes	NA	NA	NA
Option bytes(5)	0.5	Yes	Yes(3)	Yes	Yes	Yes(3)	Yes
	1	Yes	Yes(3)	Yes	Yes	Yes(3)	Yes
	2	Yes	No(6)	No	NA	NA	NA
OTP	0.5	Yes	Yes(7)	NA	Yes	Yes(7)	NA
	1	Yes	Yes(7)	NA	Yes	Yes(7)	NA
	2	Yes	Yes(7)	NA	NA	NA	NA
Backup registers	0.5	Yes	Yes	NA	Yes(2)	Yes(2)	NA(8)
	1	Yes	Yes	NA	No	No	NA(8)
	2	Yes	Yes	NA	NA	NA	NA
SRAM2	0.5	Yes	Yes	NA	Yes(2)	Yes(2)	NA(9)
	1	Yes	Yes	NA	No	No	NA(9)
	2	Yes	Yes	NA	NA	NA	NA

Table 44. Access status versus protection level and execution modes when TZEN=1

Area	RDP level	User execution (boot from Flash)			Debug/ bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
OTFDEC regions (OCTOSPI)	0.5	Yes	Yes	Yes	No(10)	Yes	Yes
	1	Yes	Yes	Yes	No(10)	Yes	Yes
	2	Yes	Yes	Yes	NA	NA	NA

1. When the protection level 2 is active, the debug port and the bootloader mode are disabled.
2. Dependent on TrustZone security access rights.
3. The Flash main memory is erased when the RDP option byte regresses from level 1 to level 0.
4. The system memory is only read-accessible whatever the protection level (0, 1 or 2) and execution mode.
5. Option bytes are only accessible through the Flash registers interface and OPSTRT bit.
6. SWAP_BANK option bit can be modified.
7. All SRAMs are erased when RDP changes from level 1 to level 0.
8. The backup registers are erased when RDP changes from level 1 to level 0.
9. All SRAMs are erased when RDP changes from level 1 to level 0.
10. The OTFDEC keys are erased when the RDP option byte changes from level 1 to level 0.

Device life cycle managed by readout protection (RDP) transitions

It is easy to move from level 0 or level 0.5 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 0.5 or level 1. Once in level 2, it is no longer possible to modify the readout protection level.

When the RDP is reprogrammed to the value 0xAA to move from level 0.5 or from level 1 to level 0, a mass erase of the Flash main memory is performed. The backup registers, all SRAMs and the OTFDEC keys are also erased. The OTP area is not erased.

When the RDP is programmed to the value 0x55 to move from level 1 to level 0.5, a partial mass erase of Flash main memory is performed. Only non-secure watermark-based areas are erased (even if it is defined as secure by block-based). The backup registers, the OTFDEC keys and all SRAMs are mass erased. The OTP area is not erased. The RDP level 0.5 and partial non-secure erase are only available when TrustZone is active.

When TrustZone is active, in case of RDP level 0.5 or level 1, the RDP regression to level 0 or level 0.5 can only be done by debug interface or by system bootloader.

Note: Full mass erase is performed only when level 1 or level 0.5 is active and level 0 requested. When the protection level is increased (0->0.5, 0->1, 0.5->1, 1->2, 0->2, 0.5->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 non-secure control register.

Note: Before launching a RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

Figure 16. RDP level transition scheme when TrustZone is disabled (TZEN=0l)

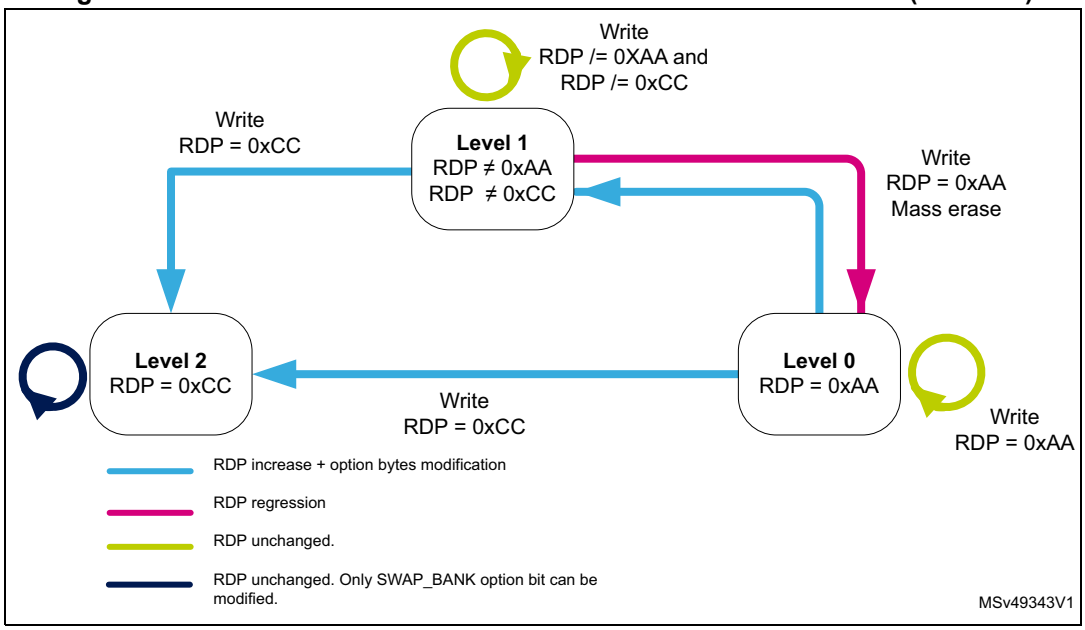
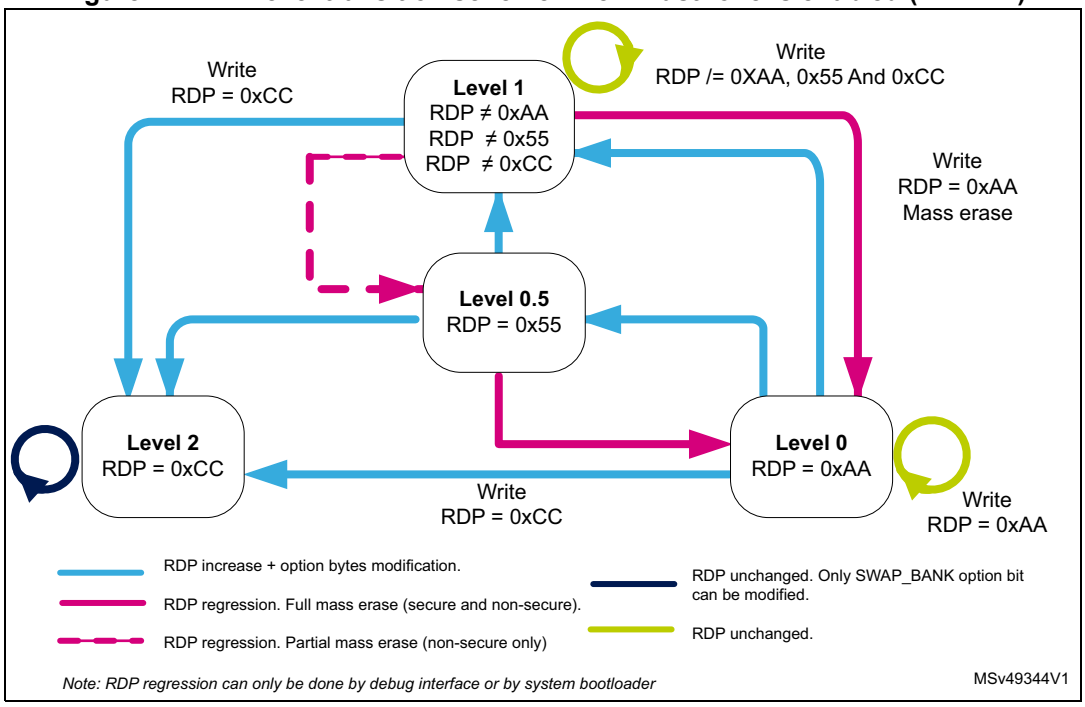


Figure 17. RDP level transition scheme when TrustZone is enabled (TZEN=1)



Summary of Flash memory and Flash registers access control

Table 45 to *Table 47* summarize all the Flash memory access versus RDP level, WPR and HDP protection when TrustZone is active and disabled.

Table 45. Flash access versus RDP level when TrustZone is active (TZEN=1)

<table>
<tr><th colspan="2" rowspan="3">Access type</th><th colspan="3">RDP level-0, RDP level-0.5, RDP level-1 no intrusion(1) or RDP level-2</th><th>RDP level-1 with intrusion(2)</th></tr>
<tr><th rowspan="2">Non-secure page</th><th colspan="2">Secure page</th><th rowspan="2">Non-secure or secure page</th></tr>
<tr><th>HDP area (HDPxEN=1 and HDPx_ACCDIS= 1)</th><th>Others(3)</th></tr>
<tr><td rowspan="4">Secure</td><td>Fetch</td><td>Bus error</td><td rowspan="2">RAZ</td><td rowspan="2">OK</td><td rowspan="2">Bus error</td></tr>
<tr><td>Read</td><td>RAZ, Flash illegal access event</td></tr>
<tr><td>Write</td><td rowspan="2">WI, SECWRPERR flag set, Flash illegal access event</td><td rowspan="2">WI, SECWRPERR flag set</td><td rowspan="2">NO WRP: OK
WRP pages: WI and SECWRPERR flag set</td></tr>
<tr><td>Page erase</td><td>WI, SECWRPERR flag set</td></tr>
<tr><td rowspan="4">Non-Secure</td><td>Fetch</td><td rowspan="2">OK</td><td colspan="2">Bus error</td><td rowspan="2">Bus error</td></tr>
<tr><td>Read</td><td colspan="2">RAZ, Flash illegal access event</td></tr>
<tr><td>Write</td><td rowspan="2">NO WRP: OK
WRP pages: WI and NSWRPERR flag set</td><td colspan="2" rowspan="2">WI, NSWRPERR flag set,
Flash illegal access event</td></tr>
<tr><td>Page erase</td><td>WI, NSWRPERR flag set</td></tr>
</table>

1. Level 1 no intrusion = when booting from user Flash and no debug access.
2. Level 1 with intrusion = when debug access is detected.
3. Others refers to other Flash memory secure configuration than the one described for HDP protections. Example: Flash secure HDP area enabled but ACCDIS=0.

Table 46. Flash access versus RDP level when TrustZone is disabled (TZEN=0)

<table>
<tr><th>Access type</th><th>RDP level-0, RDP level-1 no intrusion(1) or RDP level-2</th><th>RDP level-1 with intrusion(2)</th></tr>
<tr><td>Fetch</td><td rowspan="2">OK</td><td rowspan="3">Bus error</td></tr>
<tr><td>Read</td></tr>
<tr><td>Write</td><td rowspan="2">NO WRP: OK
WRP pages: WI and NSWRPERR flag set</td></tr>
<tr><td>Erase</td><td>WI and NSWRPERR flag set</td></tr>
</table>

1. Level 1 no intrusion = when booting from user Flash and no debug access.
2. RDP Level 1 with intrusion = when booting from user Flash and no debug access.

Table 47. Flash mass erase versus RDP level when TrustZone is active (TZEN = 1)

Access type		RDP level-0, RDP level-0.5, RDP level-1 no intrusion(1) or RDP level-2				RDP level-1 with intrusion(2)
		Non-secure Flash	Secure Flash		Mix non-secure and secure Flash	Non-secure or secure Flash
			HDP area (HDPxEN=1 and HDPx_ACCDIS = 1)	Others(3)		
Secure	Bank or mass erase	WI, SECWRPERR flag set, Flash illegal access event	WI, SECWRPERR flag set	NO WRP: OK WRP pages: WI and SECWRPERR flag set	WI, SECWRPERR flag set, Flash illegal access event	WI, SECWRPERR flag set
Non-secure	Bank or mass erase	NO WRP: OK WRP pages: WI and NSWRPERR flag set	WI, NSWRPERR flag set, Flash illegal access event			WI, NSWRPERR flag set

1. RDP Level 1 no intrusion = when booting from user Flash and no debug access.
2. RDP Level 1 with intrusion = when booting from user Flash and debug access is detected.
3. Others refers to other Flash secure configuration than the one described for HDP protections. Example: Flash secure, HDP area enabled but HDPxACCDIS = 0.

Table 48. Flash system memory, RSS and OTP accesses(1)

Access type		System memory (bootloader)	OTP	RSS
Secure	Fetch	Bus error		RAZ
	Read	RAZ, and a Flash illegal access event		
	Write	WI, SECWRPERR flag set, Flash illegal access event		WI, SECWRPERR flag set
Non-secure (TZEN = 0 or TZEN = 1)	Fetch	OK	Bus error	Bus error
	Read		OK	RAZ, Flash illegal access event(2)
	Write	WI and NSWRPERR flag set	OK if not virgin: WI, NSPROGERR flag set	WI, NSWRPERR, Flash illegal access event(3)

1. Valid for all RDP levels.
2. Flash illegal access event is only generated when TZEN=1.

Table 49. Flash registers access

Access type			Non-secure register		Secure register	
			PRIV=1	PRIV=0	PRIV=1	PRIV=0
Fetch	Secure/ non-secure	Privileged/ unprivileged	Bus error			
Read/ Write	Secure(1)	Privileged	OK			
		Unprivileged	RAZ/WI	OK	RAZ/WI	OK
	Non-secure(2)	Privileged	OK		RAZ/WI and a Flash register illegal access event(3)	
		Unprivileged	RAZ/WI	OK		

1. Secure access is only valid when TrsutZone is active (TZEN=1).
2. Non-secure access are valid when TrsutZone is active or disabled.
3. Flash register illegal access event is only generated when TZEN=1.

6.8 FLASH interrupts

Table 50. Flash interrupt request

Interrupt Vector	Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit
FLASH_S	Secure end of operation	SECEOP(1)	Write SECEOP=1	SECEOPIE
	Secure operation error	SECOPERR(2)	Write SECOPERR=1	SECERRIE
	Secure read error	SECRDERR	Write SECRDERR=1	SECRDERRIE
FLASH	Non-secure End of operation	NSEOP(3)	Write NSEOP=1	NSEOPIE
	Non-secure operation error	NSOPERR(4)	Write NSOPERR=1	NSERRIE
	ECC correction	ECCC	Write ECCC=1	ECCCIE

1. SECEOP or NSEOP are set only if SECEOPIE or is NSEOPIE is set.
2. SECOPERR is set only if SECERRIE is set.
3. SECEOP is set only if SECEOPIE is set.
4. NSOPERR is set only if NSERRIE is set.

6.9 FLASH registers

6.9.1 Flash access control register (FLASH_ACR)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x00

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
LVEN	SLEEP_PD	RUN_PD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LATENCY[3:0]			
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **LVEN**: Flash low-voltage enable

This bit can only be written when it is unlocked by writing the FLASH_LVEKEYR register. When set, this bit enable the flash low voltage and bypass the voltage range selected by PWR. This bit must be set when using the external SMPS.

When this bit is cleared, it is locked again.

0: Flash low voltage is disabled. Flash low voltage is managed by the power controller PWR.

1: Flash low voltage is enabled.

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.

Bits 12:4 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

6.9.2 Flash power-down key register (FLASH_PDKEYR)

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

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[31:0]**: 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

6.9.3 Flash non-secure key register (FLASH_NSKEYR)

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

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
NSKEYR[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
NSKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **NSKEYR[31:0]**: Flash non secure key

The following values must be written consecutively to unlock the FLACH_NSCR register allowing Flash non-secure programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

6.9.4 Flash secure key register (FLASH_SECKEYR)

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

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
SECKEYR[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
SECKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SECKEYR[31:0]**: Flash secure key

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

6.9.5 Flash option key register (FLASH_OPTKEYR)

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x10

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[31:0]**: 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

6.9.6 Flash low voltage key register (FLASH_LVEKEYR)

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x14

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
LVEKEYR[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
LVEKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **LVEKEYR[31:0]**: Flash low voltage key

The following values must be written consecutively to unlock the LVEN bit in FLASH_ACR register:
LVEKEY1: 0xF4F5 F6F7h
LVEKEY2: 0x0A1B 2C3D

6.9.7 Flash status register (FLASH_NSSR)

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

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.	NSBSY
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OPTWERR	Res.	Res.	Res.	Res.	Res.	NSPGSERR	NSSIZERR	NSPGAERR	NSWRPERR	NSPROGERR	Res.	NSOPERR	NSEOP
		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 **NSBSY**: Busy

This indicates that a Flash secure or non-secure 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 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **OPTWERR**: Option write error

Set by hardware when the options bytes are written with an invalid configuration.
Cleared by writing 1.
Refer to *Section 6.3.8: Flash errors flags* for full conditions of error flag setting.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **NSPGSERR**: Non-secure programming sequence error

This bit is set by hardware when programming sequence is not correct. It is cleared by writing 1.
Refer to *Section 6.3.8: Flash errors flags* for full conditions of error flag setting.

Bit 6 **NSSIZERR**: Non-secure size error

Set by hardware when the size of the access is a byte or half-word during a non-secure program sequence. Only double word programming is allowed (consequently: word access).
Cleared by writing 1.

Bit 5 **NSPGAERR**: Non-secure programming alignment error

Set by hardware when the first word to be programmed is not aligned with a double word address, or the second word does not belong to the same double word address.
Cleared by writing 1.

Bit 4 **NSWRPERR**: Non-secure write protection error

Set by hardware when an non-secure address to be erased/programmed belongs to a write-protected part (by WRP, HDP or RDP level 1) of the Flash memory.

Cleared by writing 1.

Refer to *Section 6.3.8: Flash errors flags* for full conditions of error flag setting.

Bit 3 **NSPROGERR**: Non-secure programming error

Set by hardware when a non-secure double-word address to be programmed contains a value different from '0xFFFF FFFF' before programming, or when already fully programmed to '0x0000 0000'.

Cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **NSOPERR**: Non-secure operation error

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

This bit is set only if non-secure error interrupts are enabled (NSERRIE = 1).

Cleared by writing ‘1’.

Bit 0 **NSEOP**: Non-secure End of operation

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

This bit is set only if the non-secure end of operation interrupts are enabled (NSEOPIE = 1).

Cleared by writing 1.

6.9.8 Flash status register (FLASH_SECSR)

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x24

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.	SECBSY
															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.	SECPGSERR	SECSIZERR	SECPGAERR	SECWRPERR	SECPROGERR	Res.	SECOPERR	SECEOP
								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 **SECBSY**: Busy

This indicates that a Flash secure or non-secure 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 Reserved, must be kept at reset value.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **SECPGSERR**: Secure programming sequence error

Set by hardware when a NSSTRT bit is set by a secure access. Set also by hardware when SECPROGERR, SECSIZERR, SECPGAERR, SECWRPERR or SECPGSER is set due to a previous programming error.
Cleared by writing 1.
Refer to *Section 6.3.8: Flash errors flags* for full conditions of error flag setting.

Bit 6 **SECSIZERR**: Secure size error

Set by hardware when the size of the access is a byte or half-word during a secure program sequence. Only double word programming is allowed (consequently: word access).
Cleared by writing 1.

Bit 5 **SECPGAERR**: Secure programming alignment error

Set by hardware when the first word to be programmed is not aligned with a double word address, or the second word does not belong to the same double word address.
Cleared by writing 1.

Bit 4 **SECWRPERR**: Secure write protection error

Set by hardware when an secure address to be erased/programmed belongs to a write-protected part (by WRP, HDP or RDP level 1) of the Flash memory.
Cleared by writing 1.
Refer to *Section 6.3.8: Flash errors flags* for full conditions of error flag setting.

Bit 3 **SECPROGERR**: Secure programming error

Set by hardware when a secure double-word address to be programmed contains a value different from '0xFFFF FFFF' before programming, or already fully programmed to '0x0000 0000'.
Cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SECOPERR**: Secure operation error

Set by hardware when a Flash memory secure operation (program / erase) completes unsuccessfully.
This bit is set only if secure error interrupts are enabled (SECERRIE = 1).
Cleared by writing ‘1’.

Bit 0 **SECEOP**: Secure end of operation

Set by hardware when one or more Flash memory secure operation (programming / erase) has been completed successfully.
This bit is set only if the secure end of operation interrupts are enabled (SECEOPIE = 1).
Cleared by writing 1.

6.9.9 Flash non-secure control register (FLASH_NSCR)

This register can only be written when the SECBSY, NSBSY or OBL_LAUNCH are reset. Otherwise, the write access is stalled till SECBSY and NSBSY are reset.

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x28

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
NSLOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	NSERRIE	NSEOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	NSSTRT
rs	rs			rc_w1		rw	rw							rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSMER2	Res.	Res.	Res.	NSBKER	Res.	NSPNB[6:0]							NSMER1	NSPER	NSPG
rw				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **NSLOCK:** FLASH_NSCR Lock

This bit is set only. When set, the FLASH_NSCR register is locked. It is cleared by hardware after detecting the unlock sequence in FLASH_NSKEYR register.
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_OPTCR register This bit is cleared by hardware after detecting the unlock sequence. The NSLOCK 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.

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

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 Reserved, must be kept at reset value.

Bit 25 **NSERRIE**: Non-secure error interrupt enable

This bit enables the interrupt generation when the NSOPERR bit in the FLASH_NSSR is set to 1.
0: NSOPERR error interrupt disabled
1: NSOPERR error interrupt enabled

Bit 24 **NSEOPIE**: Non-secure End of operation interrupt enable
This bit enables the interrupt generation when the NSEOP bit in the FLASH_NSSR is set to 1.
0: NSEOP Interrupt disabled
1: NSEOP Interrupt enabled

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **OPTSTRT**: Options modification start
This bit triggers an options operation when set. It can not be written if OPTLOCK bit is set.
This bit is set only by software, and is cleared when the NSBSY bit is cleared in FLASH_NSSR.

Bit 16 **NSSTRT**: Non-secure start
This bit triggers a non-secure erase operation when set. If NSMER1, NSMER2 and NSPER bits are reset and the NSSTRT bit is set, the NSPGERR is set. This condition should be forbidden.
This bit is set only by software, and is cleared when the NSBSY bit is cleared in FLASH_NSSR.

Bit 15 **NSMER2**: Non-secure Bank 2 Mass erase
This bit triggers the bank 2 non-secure mass erase (all bank 2 user pages) when set.

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

Bit 11 **NSBKER**: Non-secure page number MSB (bank selection)
This bit must be only set when DBANK=1.
0: Bank 1 is selected for non-secure page erase
1: Bank 2 is selected for non-secure page erase.

Bit 10 Reserved, must be kept at reset value.

Bits 9:3 **NSPNB[6:0]**: Non-secure page number selection
These bits select the page to erase:
00000000:page 0
00000001:page 1
...
11111111:page 127

Bit 2 **NSMER1**: Non-secure bank 1 mass erase
This bit triggers the bank 1 non-secure mass erase (all bank 1 user pages) when set.

Bit 1 **NSPER**: Non-secure page erase
0: Non-secure page erase disabled
1: Non-secure page erase enabled

Bit 0 **NSPG**: Non-secure programming
0: Non-secure Flash programming disabled
1: Non-secure Flash programming enabled

6.9.10 Flash secure control register (FLASH_SECCR)

This register can only be written when the SECBSY, NSBSY or OBL_LAUNCH are reset. Otherwise, the write access stalls till SECBSY and NSBSY are reset.

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x2C

Reset value: 0x8000 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
SECLOCK	Res.	SECINV	Res.	Res.	Res.	SECERRIE	SECEOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECSTRT
rs		rs				rw	rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SECMER2	Res.	Res.	Res.	SECBKER	Res.	SECPNB[6:0]							SECMER1	SECPER	SECPG
rw				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SECLOCK:** FLASH_SECCR Lock

This bit is set only. When set, the FLASH_SECCR register is locked. It is cleared by hardware after detecting the unlock sequence in FLASH_SECKEYR register.
In case of an unsuccessful unlock operation, this bit remains set until the next system reset.

Bit 30 Reserved, must be kept at reset value.

Bit 29 **SECINV:** Flash security state invert

This bit inverts the flash security state

Bits 28:26 Reserved, must be kept at reset value.

Bit 25 **SECERRIE**: Secure error interrupt enable

This bit enables the interrupt generation when the SECOPERR bit in the FLASH_SECSR is set to 1.
0: SECOPERR error interrupt disabled
1: SECOPERR error interrupt enabled

Bit 24 **SECEOPIE**: Secure End of operation interrupt enable

This bit enables the interrupt generation when the SECEOP bit in the FLASH_SECSR is set to 1.
0: SECEOP Interrupt disabled
1: SECEOP Interrupt enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **SECSTRT**: Secure start

This bit triggers a non-secure erase operation when set. If SECMER1, SECMER2 and SECPER bits are reset and the SECSTRT bit is set, the SECPGERR is set. This condition should be forbidden.
This bit is set only by software, and is cleared when the SECBSY bit is cleared in FLASH_SECSR.

Bit 15 **SECMER2**: Secure bank 2 mass erase
This bit triggers the bank 2 secure mass erase (all bank 2 user pages) when set.

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

Bit 11 **SECBKER**: Secure page number MSB (bank selection)
This bit must be only set when DBANK=1.
0: Bank 1 is selected for secure page erase
1: Bank 2 is selected for secure page erase
When DBANK=0, this bit must be kept cleared.

Bit 10 Reserved, must be kept at reset value.

Bits 9:3 **SECPNB[6:0]**: Secure page number selection
These bits select the page to erase:
00000000:page 0
00000001:page 1
...
11111111:page 127

Bit 2 **SECMER1**: Secure bank 1 mass erase
This bit triggers the bank 1 secure mass erase (all bank 1 user pages) when set.

Bit 1 **SECPER**: Secure page erase
0: Secure page erase disabled
1: Secure page erase enabled

Bit 0 **SECPG**: Non-secure programming
0: Secure Flash programming disabled
1: Secure Flash programming enabled

6.9.11 Flash ECC register (FLASH_ECCR)

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x30

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.	ECCIE	Res.	SYSF_ECC	BK_ECC	Res.	Res.	ADDR_ECC[18:16]		
rc_w1	rc_w1	rc_w1	rc_w1				rw		rw	rw			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=0

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=1

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

Set by hardware when one ECC error has been detected and corrected. An interrupt is generated if ECCIE is set.

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 ECCIE 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 **ECCIE**: 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[18:0]**: ECC fail address
This bit indicates which address is concerned by the ECC error correction or by the double ECC error detection.

6.9.12 Flash option register (FLASH_OPTR)

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x40

ST production value: 0x7FEFF8AA (Register bits 0 to 31 are loaded with values from Flash memory at OBL)

Access: no wait state when no option bytes modification 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
TZEN	Res.	Res.	PA15_PUPEN	nBOOT0	nSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	DBANK	DB256K	SWAP_BANK	WWDG_SW	IWDG_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 **TZEN:** Global TrustZone security enable
0: Global TrustZone security disabled.
1: Global TrustZone security enabled.

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **PA15_PUPEN:** PA15 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

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 PH3/BOOT0 pin

Bit 25 **SRAM2_RST:** SRAM2 erase when system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 is not erased when a system reset occurs

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

Bit 23 Reserved, must be kept at reset value.

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 all protection (secure, HDP) are disabled.

Bit 21 **DB256K:** Dual-bank on 256 Kbytes Flash memory devices
0: 256 Kbytes single Flash: contiguous address in bank1
1: 256 Kbytes dual-bank Flash with contiguous addresses.

Bit 20 **SWAP_BANK:** Swap banks
It must be only used in dual-bank mode (DBANK=1). It can only be written when all the Flash memory is non-secure. Otherwise the OPTWERR is set.
0: Bank 1 and bank 2 address are not swapped.
1: Bank 1 and bank 2 address are swapped.

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 **IWDG_SW:** Independent watchdog selection
0: Hardware independent watchdog
1: Software independent watchdog

Bit 15 Reserved, must be kept at reset value.

Bit 14 **nRST_SHDW:**
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.

Bits 10:8 **BOR_LEV[2:0]:** 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[7:0]:** Readout protection level

0xAA: Level 0, readout protection not active
0x55: Level 0.5, readout protection not active, only non-secure debug access is possible. Only available when TrustZone is active (TZEN=1)
0xCC: Level 2, chip readout protection active
Others: Level 1, memories readout protection active

Note: Refer to Section : Level 1: readout protection for more details.

6.9.13 Flash non-secure boot address 0 register (FLASH_NSBOOTADD0R)

This register can not be written if OPTLOCK bit is set.

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x44

ST production value: 0x0800007F (The option bytes are loaded with values from the Flash memory at reset release)

Access: no wait state when no option bytes modification 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
NSBOOTADD0[24:9]															
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
NSBOOTADD0[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w	w							

Bits 31:7 **NSBOOTADD0[24:0]**: Non-secure Boot base address 0

The non-secure boot memory address can be programmed to any address in the valid address range with a granularity of 128 bytes.
The NSBOOTADD0[24:0] correspond to address [31:7]. The NSBOOTADD0 option bytes are selected following the Boot pin or nSWBOOTO state.
Example:
NSBOOTADD0[24:0] = 0x0100000: Boot from non-secure Flash (0x0800 0000)
NSBOOTADD0[24:0] = 0x017F200: Boot from system memory bootloader (0x0BF9 0000)
NSBOOTADD0[24:0] = 0x0040000: Boot from non-secure SRAM1 on S-Bus(0x2000 0000)

Bits 6:0 Reserved, must be kept at reset value.

6.9.14 Flash non-secure boot address 1 register (FLASH_NSBOOTADD1R)

This register can not be written if OPTLOCK bit is set.

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x48

ST production value: 0x0BF9007F (The option bytes are loaded with values from the Flash memory at reset release)

Access: no wait state when no option bytes modification 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
NSBOOTADD1[24:9]															
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
NSBOOTADD1[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w	w							

Bits 31:7 **NSBOOTADD1[24:0]**: Non-secure boot address 1

The non-secure boot memory address can be programmed to any address in the valid address range with a granularity of 128 bytes.
The NSBOOTADD1[24:0] correspond to address [31:7]. The NSBOOTADD0 option bytes are selected following the boot pin or nSWBOOTO state.
Example:
NSBOOTADD1[24:0] = 0x0100000: Boot from non-secure Flash (0x0800 0000)
NSBOOTADD1[24:0] = 0x017F200: Boot from system memory bootloader (0x0BF9 0000)
NSBOOTADD1[24:0] = 0x0040000: Boot from non-secure SRAM1 on S-Bus(0x2000 0000)

Bits 6:0 Reserved, must be kept at reset value.

6.9.15 Flash secure boot address 0 register (FLASH_SECBOOTADD0R)

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x4C

ST production value: 0x0C00007C (The option bytes are loaded with values from the Flash memory at reset release)

Access: no wait state when no option bytes modification 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
SECBOOTADD0[24:9]															
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
SECBOOTADD0[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK
w	w	w	w	w	w	w	w	w							rs

Bits 31:7 **SECBOOTADD0[24:0]**: Secure boot base address 0

The secure boot memory address can be programmed to any address in the valid address range with a granularity of 128 bytes.
The SECBOOTADD0[24:0] correspond to address [31:7] The SECBOOTADD0 option bytes are selected following the boot pin or nSWBOOTO state.
Example:
SECBOOTADD0[24:0] = 0x018 0000: Boot from secure Flash (0x0C00 0000)
SECBOOTADD0[24:0] = 0x01F F000: Boot from RSS (0x0FF8 0000)
SECBOOTADD0[24:0] = = 0x060 0000: Boot from secure SRAM1 on S-Bus (0x3000 0000)

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **BOOT_LOCK**: BOOT LOCK

When set, the boot is always forced to base address value programmed in SECBOOTADD0[24:0] option bytes whatever the boot selection option.
When set, it cannot be cleared.

6.9.16 Flash bank 1 secure watermak1 register (FLASH_SECWM1R1)

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x50

ST production value: 0xFFFFFF80 (Register bits are loaded with values from Flash memory at OBL. Reserved bits are read as “1”)

Access: no wait state when no option bytes modification 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.	SECWM1_PEND[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.	SECWM1_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **SECWM1_PEND[6:0]:** End page of first secure area
DBANK=1
SECWM1_PEND contains the last page of the secure area in bank 1.
DBANK=0
SECWM1_PEND contains the last page of the first secure area for all memory.

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

Bits 6:0 **SECWM1_PSTRT[6:0]:** Start page of first secure area
DBANK=1
SECWM1_PSTRT contains the first page of the secure area in bank 1.
DBANK=0
SECWM1_PSTRT contains the first page of the first secure area for all memory.

6.9.17 Flash secure watermak1 register 2 (FLASH_SECWM1R2)

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x54

ST production value: 0x7F807F80 (Register bits are loaded with values from Flash memory at OBL)

Access: no wait state when no option bytes modification 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
HDP1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_PEND[6:0]						
rw									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **HDP1EN**: Hide protection first area enable
0: No HDP area 1
1: HDP first area is enabled.

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **HDP1_PEND[6:0]**: End page of first hide protection area
DBANK=1
HDP1_PEND contains the last page of the HDP area in bank1.
DBANK=0
HDP1_PEND contains the last page of the first HDP area for all memory.

Bits 15:0 Reserved, must be kept at reset value.

6.9.18 Flash WPR1 area A address register (FLASH_WRP1AR)

This register can not be written if OPTLOCK bit is set.

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x58

ST production value: 0xFF80FFFF (Register bits are loaded with values from Flash memory at OBL. Reserved bits are read as “1”)

Access: no wait state when no option bytes modification 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_PEND[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_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **WRP1A_PEND[6:0]**: Bank 1 WPR first area “A” end page
DBANK=1
WRP1A_PEND contains the last page of the first WPR area in bank 1.
DBANK=0
WRP1A_PEND contains the last page of the first WPR area for all memory.

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

Bits 6:0 **WRP1A_PSTRT[6:0]**: bank 1 WPR first area “A” start page
DBANK=1
WRP1A_PSTRT contains the first page of the first WPR area for bank1.
DBANK=0
WRP1A_PSTRT contains the first page of the first WPR area for all memory.

6.9.19 Flash WPR1 area B address register (FLASH_WRP1BR)

This register can not be written if OPTLOCK bit is set.

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x5C

ST production value: 0xFF80FFFF (Register bits are loaded with values from Flash memory at OBL)

Access: no wait state when no option bytes modification 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_PEND[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_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **WRP1B_PEND[6:0]**: Bank 1 WRP second area “B” end page
DBANK=1
WRP1B_PEND contains the last page of the second WRP area in bank1.
DBANK=0
WRP1B_PEND contains the last page of the second WRP area for all memory.

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

Bits 6:0 **WRP1B_PSTRT[6:0]**: Bank 1 WRP second area “B” start page
DBANK=1
WRP1B_PSTRT contains the first page of the second WRP area for bank1.
DBANK=0
WRP1B_PSTRT contains the first page of the second WRP area for all memory.

6.9.20 Flash secure watermak2 register (FLASH_SECWM2R1)

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x60

ST production value: 0xFFFFFF80 (Register bits are loaded with values from Flash memory at OBL)

Access: no wait state when no option bytes modification 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.	SECWM2_PEND[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.	SECWM2_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **SECWM2_PEND[6:0]**: End page of second secure area
DBANK=1
SECWM2_PEND contains the last page of the secure area in bank 2.
DBANK=0
SECWM2_PEND contains the last page of the second secure area for all memory.

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

Bits 6:0 **SECWM2_PSTRT[6:0]**: Start page of second secure area
DBANK=1
SECWM2_PSTRT contains the first page of the secure area in bank 2.
DBANK=0
SECWM2_PSTRT contains the first page of the second secure area for all memory.

6.9.21 Flash secure watermak2 register 2 (FLASH_SECWM2R2)

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x64

ST production value: 0x7F807F80 (Register bits are loaded with values from Flash memory at OBL)

Access: no wait state when no option bytes modification 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
HDP2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_PEND[6:0]						
rw									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **HDP2EN:** Hide protection second area enable
0: No HDP area 2
1: HDP second area is enabled.

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **HDP2_PEND[6:0]**: End page of hide protection second area
DBANK=1
HDP2_PEND contains the last page of the HDP area in bank 2.
DBANK=0
HDP2_PEND contains the last page of the second HDP area for all memory.

Bits 15:0 Reserved, must be kept at reset value.

6.9.22 Flash WPR2 area A address register (FLASH_WRP2AR)

This register can not be written if OPTLOCK bit is set.

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x68

ST production value: 0xFF80FFFF (Register bits are loaded with values from Flash memory at OBL)

Access: no wait state when no option bytes modification 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_PEND[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_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **WRP2A_PEND[6:0]**: Bank 2 WPR first area “A” end page
DBANK=1
WRP2A_PEND contains the last page of the first WRP area in bank2.
DBANK=0
WRP2A_PEND contains the last page of the third WRP area for all memory.

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

Bits 6:0 **WRP2A_PSTRT[6:0]**: Bank 2 WPR first area “A” start page
DBANK=1
WRP2A_PSTRT contains the first page of the first WRP area for bank2.
DBANK=0
WRP2A_PSTRT contains the first page of the third WRP area for all memory.

6.9.23 Flash WPR2 area B address register (FLASH_WRP2BR)

This register can not be written if OPTLOCK bit is set.

This register is non-secure. It can be read and written by both secure and non-secure access.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x6C

ST production value: 0xFF80FFFF (Register bits are loaded with values from Flash memory at OBL)

Access: no wait state when no option bytes modification 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_PEND[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.	WRP2B_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **WRP2B_PEND[6:0]**: Bank 2 WPR second area “B” end page
DBANK=1
WRP2B_PEND contains the last page of the second WRP area in bank 2.
DBANK=0
WRP2B_PEND contains the last page of the fourth WRP area for all memory.

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

Bits 6:0 **WRP2B_PSTRT[6:0]**: Bank 2 WPR second area “B” start page
DBANK=1
WRP2B_PSTRT contains the first page of the second WRP area for bank 2.
DBANK=0
WRP2B_PSTRT contains the first page of the fourth WRP area for all memory.

6.9.24 FLASH secure block based bank 1 register (FLASH_SECBB1Rx) (where x=1..4)

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0x80 + 4 *(x - 1), (x=1..4)

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification 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
SECBB1[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
SECBB1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECBB1[31:0]:** page secure/non-secure attribution

This bit is used to set the page security attribution in bank 1 when dual bank mode or for the all memory in single bank mode.
DBANK=1
0:Page (32 * x+y) in bank 1 is non secure
1:Page (32 * x+y) i in bank 1 is secure
DBANK=0
0:Page (32 * x+y) in all memory is non secure
1:Page (32 * x+y) in all memory is secure

6.9.25 FLASH secure block based bank 2 register (FLASH_SECBB2Rx) (where x=1..4)

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0xA0 + 4 *(x - 1), (x=1..4)

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification 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
SECBB2[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
SECBB2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECBB2[31:0]:** page secure/non-secure attribution

This bit is used to set the page security attribution in bank 2. This must be only used in dual bank mode. In single bank mode, writing this bit has no effect and written data is ignored.
DBANK=1
0:Page (32 * x+y) in bank 2 is non secure
1:Page (32 * x+y) i in bank 2 is secure
DBANK=0
Reserved, must be kept at reset value.

6.9.26 FLASH secure HDP control register (FLASH_SECHDPCR)

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the FLASH_PRIVCFGR register.

Address offset: 0xC0

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification 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.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_ACCDIS	HDP1_ACCDIS
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HDP2_ACCDIS:** HDP2 area access disable

When set, this bit is only cleared by a system reset
0:Access to HDP2 area is granted
1:Access to HDP2 area is denied and options bytes modification listed in *Table 34: Default secure option bytes after TZEN activation* are denied

Bit 0 **HDP1_ACCDIS:** HDP1 area access disable

When set, this bit is only cleared by a system reset
0:Access to HDP1 area is granted
1:Access to HDP1 area is denied and options bytes modification listed in *Table 34: Default secure option bytes after TZEN activation* are denied

6.9.27 FLASH privilege configuration register (FLASH_PRIVCFGR)

This register can be read by both privileged and unprivileged access.

When the system is secure (TZEN =1),this register can be read by secure and non-secure access. It is write-protected against non-secure write access when the Flash is secure. A non-secure write access is ignored and generates an illegal access event.

Address offset: 0xC4

Reset value: 0x0000 0000

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PRIV**: Privilege protection

This bit can be read by both privileged or unprivileged, secure and non-secure access. when set, it can only be cleared by a privileged access.

0: All Flash registers can be read and written by privileged or unprivileged access.

1: All Flash registers can be read and written by privileged access only.

If the Flash is not secure (no secure area defined), the PRIV bit can be written by a secure or non-secure privileged access.

If the Flash is secure, the PRIV bit can be written only by a secure privileged access:

– A non-secure write access is ignored and generates an illegal access event.

– A secure unprivileged write access on PRIV bit is ignored

6.9.28 FLASH register map and reset values

Table 51. 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.	Res.	Res.	Res.	LVEN	SLEEP_PD	RUN_PD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LATENCY [3:0]			
	Reset value																	0	0	0										0	0	0	0
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_NSKEYR	NSKEYR[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_SECKEYR	SECKEYR[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_OPTKEYR	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
0x14	FLASH_LVEKEYR	LVEKEYR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	FLASH_NSSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSBSY	Res.	Res.	Res.	OPTWERR	Res.	Res.	Res.	Res.	NSPGSERR	NSSIZERR	NSPGAERR	NSWRPERR	NSPROGERR	Res.	NSOPERR	NSEOP
	Reset value																0				0					0	0	0	0	0		0	0

Table 51. 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
0x24	FLASH_SECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECBSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECPGSERR	SECSIZERR	SECPGAERR	SECWRPERR	SECPROGERR	Res.	SECOPERR	SECEOP
	Reset value																0									0	0	0	0	0		0	0
0x28	FLASH_NSCR	NSLOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	NSERRIE	NSEOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	NSSTRT	NSMER2	Res.	Res.	Res.	NSBKER	Res.	NSPNB[6:0]						NSMER1	NSPER	NSPG
	Reset value	1	1			0		0	0							0	0	0				0				0	0	0	0	0	0	0	0
0x2C	FLASH_SECCR	SECLOCK	Res.	SECINV	Res.	Res.	Res.	SECERRIE	SECEOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECSTRT	SECMER2	Res.	Res.	Res.	SECBKER	SECPNB[7:0]							SECMER1	SECPER	SECPG
	Reset value	1		0				0	0								0	0				0	0							0	0	0
0x30	FLASH_ECCR	ECCD	ECCC	ECCD2	ECCC2	Res.	Res.	Res.	ECCIE	Res.	SYSF_ECC	BK_ECC	ADDR_ECC[20:0]																				
	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
0x40	FLASH_OPTR	TZEN	Res.	Res.	PA15_PUPEN	nBOOT0	nSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	DBANK	DB256K	SWAP_BANK	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW	Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
	ST production value	0			1	1	1	1	1		1	1	0	1	1	1	1		1	1	1		0	0	0	1	0	1	0	1	0	1	0
0x44	FLASH_NSBOOTADD0R	NSBOOTADD0[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	ST production value	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x48	FLASH_NSBOOTADD1R	NSBOOTADD1[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	ST production value	0	0	0	0	1	0	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0							
0x4C	FLASH_SECBOOTADD0R	SECBOOTADD0[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK
	ST production 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
0x50	FLASH_SECWM1R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_PSTRT[6:0]						
	ST production value										1	1	1	1	1	1	1										0	0	0	0	0	0	0
0x54	FLASH_SECWM1R2	HDP1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	ST production value	1									0	0	0	0	0	0	0																
0x58	FLASH_WRP1AR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_PSTRT[6:0]						
	ST production value										0	0	0	0	0	0	0										1	1	1	1	1	1	1

Table 51. 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
0x5C	FLASH_WRP1BR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_PSTRT[6:0]						
	ST production value										0	0	0	0	0	0	0										1	1	1	1	1	1	1
0x60	FLASH_ SECWM2R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_PSTRT[6:0]						
	ST production value										1	1	1	1	1	1	1										0	0	0	0	0	0	0
0x64	FLASH_ SECWM2R2	HDP2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	ST production value	1									0	0	0	0	0	0	0															
0x68	FLASH_WRP2AR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_PSTRT[6:0]						
	ST production value										0	0	0	0	0	0	0										1	1	1	1	1	1	1
0x6C	FLASH_WRP2BR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_PSTRT[6:0]						
	ST production value										0	0	0	0	0	0	0										1	1	1	1	1	1	1
0x80 + 4 *(x - 1), (x=1..4)	FLASH_ SECBB1Rx	SECBB1[y]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0 + 4 *(x - 1), (x=1..4)	FLASH_ SECBB2Rx	SECBB2[y]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC0	FLASH_ SECHDPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_ ACCDIS	HDP1_ ACCDIS
	Reset value																															0	0
0xC4	FLASH_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV
	Reset value																																0

Refer to *Section 2.3 on page 87* for the register boundary addresses.

7 Instruction cache (ICACHE)

7.1 Introduction

The instruction cache (ICACHE) is introduced on C-AHB code bus of Cortex-M33 processor to improve performance when fetching instruction and data from both internal and external memories.

Some specific features like dual master port, hit-under-miss and critical-word-first refill policy, allow close to zero wait states performance in most use cases.

7.2 ICACHE main features

The main features of ICACHE are described below:

- Bus interface
  - one 32-bit AHB slave port, the execution port (input from Cortex-M33 C-AHB code interface)
  - two 32-bit AHB master ports: master1 and master2 ports (outputs to Fast and Slow buses of main AHB busmatrix, respectively)
  - one 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Dual master access: feature used to decouple the traffic according to targeted memory. For example, ICACHE assigns fast traffic (addressing FLASH and SRAM memories) to the AHB master1 port, and slow traffic (addressing external memories sitting on OCTOSPI and FMC interfaces) to AHB master2 port, thus preventing processor stalls on lines refills from external memories. This allows ISR (interrupt service routine) fetching on internal FLASH memory to take place in parallel with a cache line refill from external memory.
  - Minimal impact on interrupt latency, thanks to dual master
  - Optimal cache line refill thanks to WRAP bursts of the size of the cache line (such as WRAP4 for 128-bit cache line)
  - 2-ways set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile
- Memory address remap
  - Possibility to remap input address falling into up to four memory regions (used to remap aliased code in external memories to the internal Code region, for execution)
- Replacement and refill
  - pLRU-t replacement policy (pseudo-least-recently-used, based on binary tree), algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls

  - Possibility to configure burst type of AHB memory transaction for remapped regions: INCRw or WRAPw (size w aligned on cache line size)
- Performance counters

  ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)
- Error management
  - Possibility to detect an unexpected cacheable write access, to flag an error and optionally to raise an interrupt
- TrustZone® security support
- Maintenance operation
  - Cache invalidate: full cache invalidation, fast command, non interruptible

7.3 ICACHE implementation

Table 52. ICACHE features

Feature	ICACHE
Number of ways	2
Cache size	8 Kbytes
Cache line width	16 bytes
range granularity of memory regions to be remapped	2 Mbytes
Number of regions to remap	4
Data size of AHB fast master1 interface	32 bits
Data size of AHB slow master2 interface	32 bits

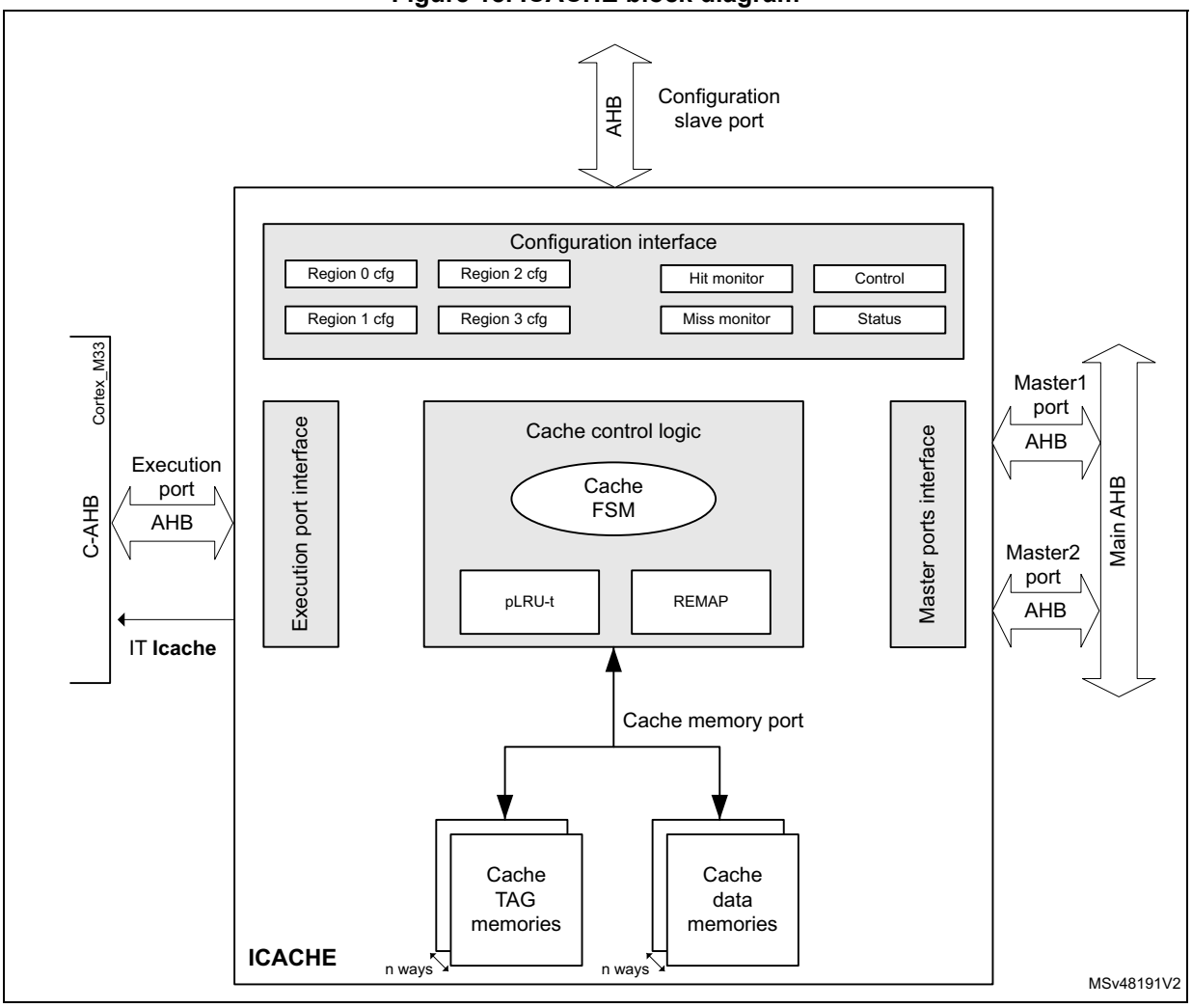
7.4 ICACHE functional description

The purpose of the instruction cache is to cache instruction fetches or instruction memories loads, coming from the processor. As such ICACHE only manages read transactions and does not manage write transactions.

For error management purpose, in case a write cacheable transaction is presented (this only happens in case of bad software programming), ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

7.4.1 ICACHE block diagram

Figure 18. ICACHE block diagram



7.4.2 ICACHE reset and clocks

ICACHE is clocked on Cortex-M33 C-AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no more busy)
- ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (=0).

Note: When disabled, ICACHE is bypassed, except the remapping mechanism that is still functional: the slave input requests (remapped or not) are just forwarded to the master port(s).

7.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags, that indicate which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits are reset in any of the below cases:

- after ICACHE reset is released
- when cache is disabled, by setting the EN bit low in the ICACHE_CR register (by software)
- when executing ICACHE invalidate command, by setting the CACHEINV bit high in the ICACHE_CR register (by software)

When a cacheable transaction is received at input execution port, its AHB address (HADDR_in) is split into the following fields (see table below for definition of B and W):

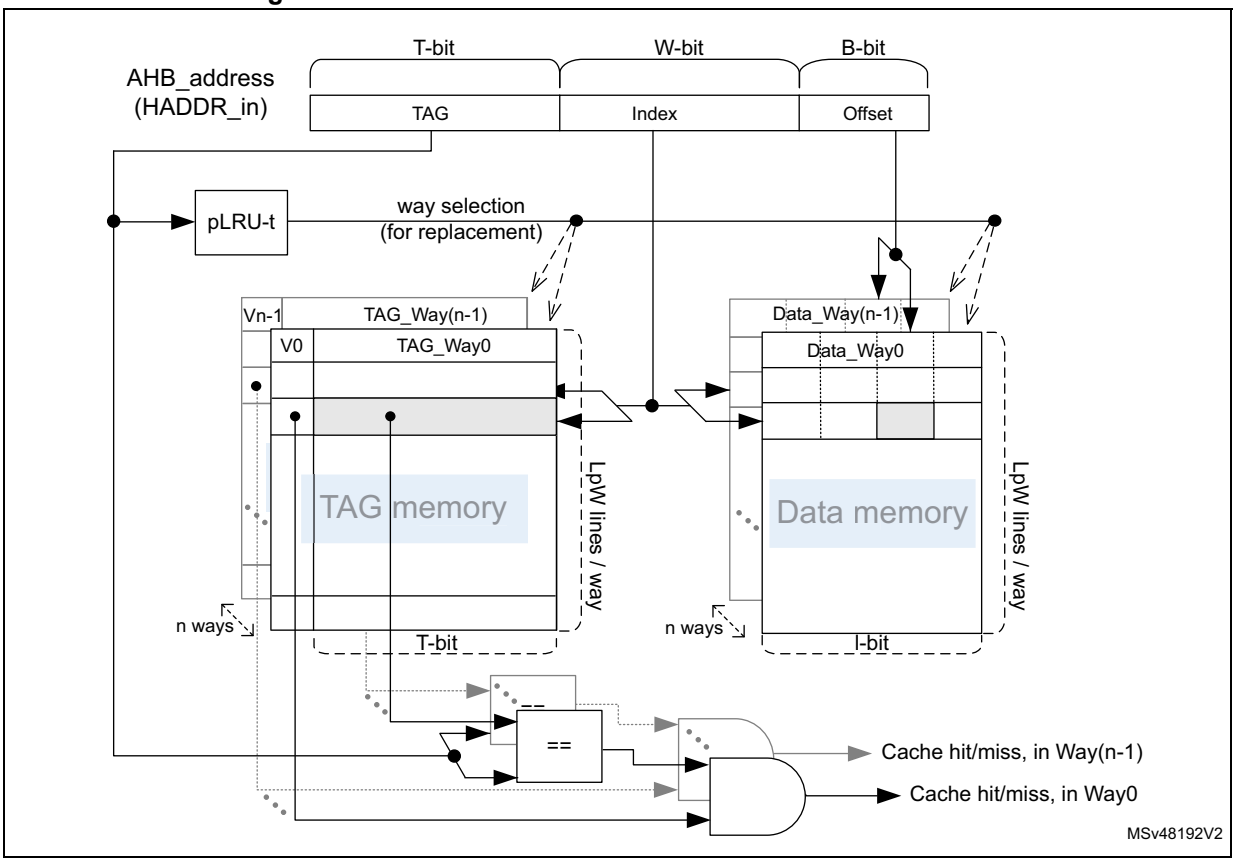
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The table below gives a summary of ICACHE main parameters for TAG memory dimensioning and *Figure 19* shows the functional view of TAG and data memories, for a n-way set associative ICACHE.

Table 53. TAG memory dimensioning parameters for n-way set associative operating mode (default)

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines (per way)	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = log2(L) bit	4-bit
Address way index size	W = log2(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 19. ICACHE TAG and data memories functional view



7.4.4 Direct mapped ICACHE (1-way cache)

Default configuration (at reset) is 2-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure ICACHE as direct mapped by writing WAYSEL = 0 (only possible when cache is disabled, EN=0 in ICACHE_CR).

The table below gives a summary of ICACHE main parameters for TAG memory in case the direct mapped cache operating mode is selected.

Table 54. TAG memory dimensioning parameters for direct mapped cache mode

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	1	1
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte=128-bit
Number of cache lines	LpW = s / L lines	512 lines
Address byte offset size	B = log2(L) bit	4-bit
Address way index size	W = log2(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	19-bit

All cache operations (such as read, refill, remapping, invalidation) remain the same in direct mapped configuration; the only difference is the absence of a replacement algorithm in case of line eviction (as explained in *Section 7.4.8*), since only one way (the unique one) is possible for any data refill.

7.4.5 ICACHE enable

In order to activate the ICACHE functioning, the EN bit must be set in the ICACHE_CR register.

When ICACHE is disabled, it is bypassed and all transactions are copied from slave port to master ports in the same clock cycle.

It is recommended to initialize or to modify the main memory content (the region to be later cached) with ICACHE disabled, and to enable ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

In order to insure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This invalidate procedure occurs when hardware reset signal is released, when ICACHE_CR.CACHEINV is set or when ICACHE_CR.EN is cleared. During the procedure, ICACHE_SR.BUSYF is set, and once finished, ICACHE_SR.BUSYF is cleared and ICACHE_SR.BSYENDF is set (raising the ICACHE interrupt if enabled on such a busy end condition).

Software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF still at 1) is treated as non cacheable, and its performance depends on the main memory access time.

The address remapping is performed, whether ICACHE is enabled or not, if input transaction address falls into memory regions defined and enabled in ICACHE_CRRx (see *Figure 20*).

ICACHE is by default disabled at boot.

7.4.6 Cacheable and non-cacheable traffic

ICACHE is developed for Cortex-M33 core. It is placed on C-AHB bus, and thus caches the Code memory region, ranging from address 0x0000 0000 to 0x1FFF FFFF of the memory map.

In order to make some other memory regions cacheable, ICACHE supports a memory region remapping feature. It allows to define up to four external memory regions, which addresses have an alias in the Code region. Addressing these external memory regions through their Code alias address allows the memory request to be routed to the C-AHB bus and to be managed by ICACHE.

Typically, any external memory space physically mapped at an address somewhere in range [0x6000 0000:0x9FFF FFFF] can be aliased with an address in range [0x0000 0000:0x07FF FFFF] or [0x1000 0000:0x1FFF FFFF].

For a given memory request in the Code region, ICACHE implements the address remapping functionality first. If aliased, it is the remapped address which is then cached, and, if needed, provided to the master port to address the main AHB busmatrix. The destination physical address does not need further manipulation on the AHB bus.

The remapping functionality is available also for non-cacheable traffic and when cache is disabled.

Further details on address remapping are provided in *Section 7.4.7*.

An incoming memory request to ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 55*. This AHB attribute depends on the MPU programming for the addressed region.

Table 55. ICACHE cacheability for AHB transaction

AHB Lookup attribute	Cacheability
1	Cacheable
0	Non cacheable

In case of non cacheable access, ICACHE is bypassed, meaning that the AHB transaction is propagated unchanged to the master output port, except the transaction address which may be modified due to the address remapping feature (see *Section 7.4.7*).

The bypass, and eventual remap logic, does not increase the latency of the access to the targeted memory.

In case of cacheable access, the ICACHE behaves as explained in *Section 7.4.8*.

Cacheable memory regions are defined and programmed by the user in the memory protection unit (MPU), that is responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

Table 56 summarizes product memories programmable configurations.

Table 56. Configurations of product memories

Product memory	Cacheable (MPU programming)	Remapped in ICACHE (ICACHE_CRRx programming)
FLASH	Yes or No	Not required
SRAM	Not recommended	Not required
External memories (OCTOSPI, FMC)	Yes	Required
	No	Required if the user wants external code fetching on C-AHB bus (else on S-AHB bus)

7.4.7 Address remapping

ICACHE allows to define an alias address in Code region for up to four external memory regions.

The address remapping is applied on the Code alias address, transforming it into the destination external physical address.

The remapping operation is fully software configurable by programming ICACHE_CRRx register (x = 0 to 3, number of remapped regions). This programming can be done only when ICACHE is disabled.

Each region x can be individually enabled with the REN bit in ICACHE_CRRx. Once enabled, the remap operation occurs even if ICACHE is disabled or if the transaction is not cacheable.

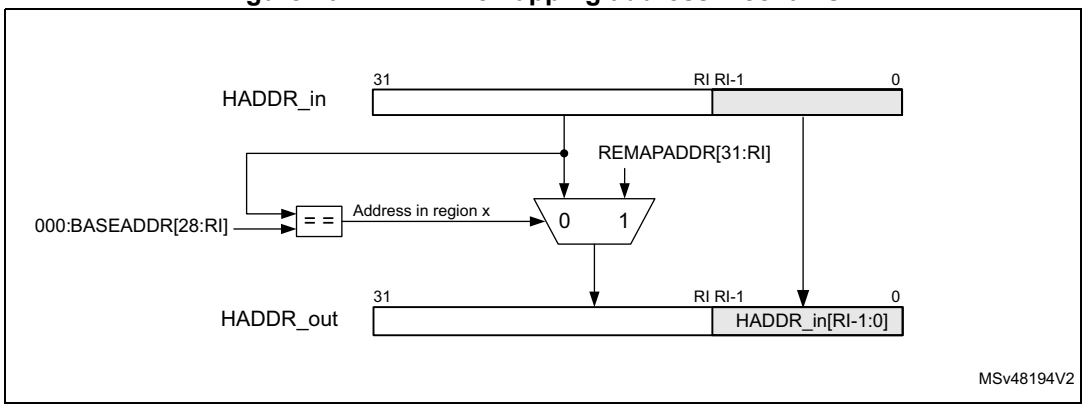
Remap regions can have different size: each region size can be programmed in the RSIZE field of its ICACHE_CRRx register. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum region size of 2 Mbytes and a maximum region size of 128 Mbytes.

The address remapping mechanism is based on the matching of an incoming AHB address (HADDR_in) with a given Code sub-region base address, and the modification of this address into its (remapped) external physical address, as follows:

- HADDR_in belongs to region x if HADDR_in[31:RI] = 000:BASEADDR[28:RI], where:
  - 000:BASEADDR is the code sub-region base address programmed in the BASEADDR field of ICACHE_CRRx.
  - RI defines the number of significant bits to consider. RI = log2(region size) with a minimum value of 21 (for a 2-Mbyte region) and a maximum value of 27 (for a 128-Mbyte region)
- If region x is enabled, the master port output AHB address (HADDR_out) is then composed by concatenating the two below parts:
  - REMAPADDR[31:RI] field of ICACHE_CRRx as MSBs
  - HADDR_in[RI-1:0] as LSBs.

The figure below describes the matching and the output address generation.

Figure 20. ICACHE remapping address mechanism



The table below summarizes all possible configurations of BASEADDR and REMAPADDR sizes (number of significant MSBs) in ICACHE_CRRx, depending on RSIZE.

Table 57. ICACHE remap region size, base address and remap address

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
2	8	11
4	7	10
8	6	9
16	5	8
32	4	7

Table 57. ICACHE remap region size, base address and remap address (continued)

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
64	3	6
128	2	5

Care must be taken while programming BASEADDR and REMAPADDR fields in ICACHE_CRRx: if programmed value is bigger than expected (in terms of number of MSBs, see *Table 57*), the unnecessary extra LSBs are ignored.

Typical remapping example: a 128-Mbyte FMC region (NOR/SRAM) physically located in the external address range [0x6800 0000:0x6FFF FFFF], remapped in Code section range [0x1000 0000:0x17FF FFFF]:

- REMAPADDR[31:21] = 0x340
- BASEADDR[28:21] = 0x80
- HADDR_in[31:27] is compared to 000:BASEADDR[28:27], and HADDR_in/BASEADDR[26:21] are ignored for the comparison.

If the comparison matches:

- HADDR_out[31:27] gets REMAPADDR[31:27] (in place of HADDR_in[31:27])
- HADDR_out[26:0] gets HADDR_in[26:0]

The software can program the kind of AHB burst that is generated by ICACHE master ports on busmatrix (for cache line refill), by setting the HBURST bit in ICACHE_CRRx with:

- WRAP for remapped external memories accessed through OCTOSPI interface that can support WRAP burst mode, providing the benefit of the critical-word-first feature performance:
  - WRAP burst size = cache line size
  - WRAP burst start address = word address of the first data requested by the core
- INCR: INCR burst mode for external memories accessed through FMC interface that does not support WRAP burst mode (losing the benefit of critical-word-first feature):
  - INCR burst size = cache line size
  - INCR burst start address = address aligned on the boundary of the cache line containing the requested word.

Note: Coherency is needed when programming SAU (secure attribution unit) and MPU (memory protection unit) attributes for both the external regions and their aliased Code sub-regions.

7.4.8 Cacheable accesses

When ICACHE receives a cacheable transaction from Cortex-M33, ICACHE checks if the address requested is present in its TAG memory and if the corresponding cache line is valid.

There are then three alternatives:

- Address is present inside the TAG memory, cache line is valid: **cache hit**, the data is read from cache and provided to the processor in the same cycle.
- Address is not present in the TAG memory: **cache miss**, the data is read from main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first policy insures minimum wait cycles for the processor, since read data can be provided while cache is still performing cache line refill (associated latency is the latency of fetching one word from main memory).

  In case no address remap occurs, the burst generated on ICACHE master bus is WRAPw (w being the cache line width, in words); if an address remap occurs, the kind of burst depends on the HBURST bit programmed in the corresponding ICACHE_CRRx register.

  The AHB transaction attributes are also propagated to main AHB busmatrix on the master port selected for the line refill.
- Address is not present in TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache hit** (hit-under-miss feature).

  This happens during cache line refill, ICACHE is capable of providing the requested data as soon as the data is available at its master interface, thus avoiding a miss (fetching data from main memory).

In case of cache refill (due to cache miss), ICACHE selects which cache line is written with the refill data:

- In direct map (1-way) mode, only one line can be used to store the refill data, the line pointed by the index of the input address.
- In n-way set associative mode, one line among 2 can be used (the line pointed by the address index, in each of the 2 ways). The way selection is based on a pLRU-t replacement algorithm. This algorithm points, for each index, on the way candidate for the next refill.

If ever the cache line where the refill data must be written, is already valid, the targeted cache line must be invalidated first; this is true whatever the direct map or n-way set associative cache mode.

7.4.9 Dual master cache

ICACHE can implement a dual port AHB master on main AHB busmatrix: master1 and master2 ports. This allows to split the traffic going to different destination memories.

The non-remapped traffic goes systematically to master1 port. The re-mapped traffic to external memories must be routed on master2 port by programming the MSTSEL bit of ICACHE_CRRx (on a region basis).

Typically, code can be fetched as follows:

- internal FLASH memory and internal SRAM on master1 port (Fast bus)
- external FLASH/RAM (through OCTOSPI/FMC interfaces) on master2 port (Slow bus)

For systems not implementing external memories, it is also possible to decouple the traffic to the internal FLASH memory from the traffic to the internal SRAM (when remapped by the ICACHE). This feature allows to prevent further processor stalls on misses.

Alongside with hit-under-miss, this dual master feature allows the processor to have an alternative path in case of fetching from different memories.

7.4.10 ICACHE security

ICACHE implements a v8-M TrustZone as defined by Arm.

ICACHE configuration registers are protected at system level.

7.4.11 ICACHE maintenance

The software can invalidate the whole content of the ICACHE by programming the CACHEINV bit in the ICACHE_CR register.

When CACHEINV is set, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, reseting each TAG valid bit to 0 (one valid bit per cache line). The CACHEINV bit of ICACHE_CR is also automatically cleared.

Once the invalidate operation is finished, ICACHE automatically clears the BUSYF flag and sets the BSYENDF flag in the ICACHE_SR register.

If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the **ICACHE** interrupt is raised.

Then, the (empty) cache is available again.

7.4.12 ICACHE performance monitoring

ICACHE provides two monitors for performance analysis: a 32-bit hit monitor and a 16-bit miss monitor.

- The hit monitor counts the AHB-transactions at the input of ICACHE (execution port) that do not generate a transaction on ICACHE output (master1 or master2 port).

  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from cache master port) (see *Section 7.4.8*)
- The miss monitor counts the AHB-transactions at the input of the ICACHE (execution port) that generate a transaction on ICACHE output (master1 or master2 port).

  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, monitors do not wrap over.

Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through the HITMEN bit in ICACHE_CR.
- Reset the hit monitor by setting the HITMRST bit in ICACHE_CR.
- Enable/stop the miss monitor through the MISSMEN bit in ICACHE_CR.
- Reset the miss monitor by setting the MISSMRST bit in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

7.4.13 ICACHE Boot

ICACHE is disabled (EN = 0 in ICACHE_CR) at Boot.

Code remapping at Boot is not needed for Cortex®-M33 since it implements the VTOR (vector tables) that allows a boot start address definition different than 0x0.

Once Boot is finished, ICACHE can be enabled (software setting the EN bit to 1 in ICACHE_CR).

7.5 ICACHE low-power modes

At product level, using ICACHE reduces the power consumption by fetching instructions from the internal ICACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with a lower-performance profile (in terms of hit ratio) and stringent low-power consumption constraints, may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see *Figure 19*). The power consumption is then reduced by accessing, for each request, only the necessary cut of TAG and data memories. Meanwhile, the cache effect still improves fetch performance; even if for most codes execution, it is a little less efficient than with an n-way set associative cache mode.

7.6 ICACHE error management and interrupts

In case an unsupported cacheable write request is detected (functional error), ICACHE generates an error by setting the ERRF flag in ICACHE_SR. In such a case, an interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is then generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

Both interrupts use the same **ICACHE** interrupt vector.

Table 58. ICACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
ICACHE	Functional error	ERRF flag in ICACHE_SR	ERRIE bit in ICACHE_IER	Set CERRF bit to 1 in ICACHE_FCR
	End of busy state (invalidate finished)	BSYENDF flag in ICACHE_SR	BSYENDIE bit in ICACHE_IER	Set CBSYENDF bit to 1 in ICACHE_FCR

ICACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from master1 or master2 port back to the execution port.

7.7 ICACHE registers

7.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN
												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.	WAYSEL	CACHEINV	EN
													rw	w	rw

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

Bit 19 **MISSMRST**: miss monitor reset
0: no effect
1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset
0: no effect
1: reset cache hit monitor

Bit 17 **MISSMEN**: miss monitor enable
0: cache miss monitor switched off. Stopping the monitor does not reset it.
1: cache miss monitor enabled

Bit 16 **HITMEN**: hit monitor enable
0: cache hit monitor switched off. Stopping the monitor does not reset it.
1: cache hit monitor enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **WAYSEL**: cache associativity mode selection
This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).
0: direct mapped cache (1-way cache)
1: n-way set associative cache (reset value)

Bit 1 **CACHEINV**: cache invalidation
Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN**: enable
0: cache disabled
1: cache enabled

7.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRF**: cache error flag
0: no error
1: an error occurred during the operation (cacheable write)

Bit 1 **BSYENDF**: busy end flag
0: cache busy
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF**: busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

7.7.3 ICACHE interrupt enable register (ICACHE_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.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRIE**: interrupt enable on cache error

Set by software to enable an interrupt generation in case of cache functional error (cacheable write access)

0: interrupt disabled on error

1: interrupt enabled on error

Bit 1 **BSYENDIE**: interrupt enable on busy end

Set by software to enable an interrupt generation at the end of a cache invalidate operation.

0: interrupt disabled on busy end

1: interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

7.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CERRF**: clear cache error flag

Set by software.

0: no effect

1: clears ERRF flag in ICACHE_SR

Bit 1 **CBSYENDF**: clear busy end flag

Set by software.

0: no effect

1: clears BSYENDF flag in ICACHE_SR.

Bit 0 Reserved, must be kept at reset value.

7.7.5 ICACHE hit monitor register (ICACHE_HMONR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **HITMON[31:0]**: cache hit monitor counter

7.7.6 ICACHE miss monitor register (ICACHE_MMONR)

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
MISSMON[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 **MISSMON[15:0]**: cache miss monitor counter

7.7.7 ICACHE region x configuration register (ICACHE_CRRx)

Address offset: 0x020 + 4 * x, (x = 0 to 3)

Reset value: 0x0000 0200

Define a Code alias address for external regions, making them cacheable. BASEDADDR and REMAPADDR fields are write locked (read only) when ICACHE_CR.EN is high.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]										
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
REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	BASEADDR[28:21]							
rw				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **HBURST**: output burst type for region x
0: WRAP
1: INCR

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **MSTSEL**: AHB cache master selection for region x
0: no action (master1 selected by default)
1: master2 selected

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **REMAPADDR[31:21]**: remapped address for region x
This field replaces the alias address defined by BASEADDR field.
The only useful bits are [31:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 7.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

Bit 15 **REN**: enable for region x
0: disabled
1: enabled

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

Bits 11:9 **RSIZE[2:0]**: size for region x
000: reserved
001: 2 Mbytes
010: 4 Mbytes
011: 8 Mbytes
100: 16 Mbytes
101: 32 Mbytes
110: 64 Mbytes
111: 128 Mbytes

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **BASEADDR[28:21]**: base address for region x
This alias address is replaced by REMAPADDR field.
The only useful bits are [28:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 7.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

7.7.8 ICACHE register map

Table 59. ICACHE 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	**ICACHE_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
	Reset value													0	0	0	0														1	0	0
0x004	**ICACHE_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.	ERRF	BSYENDF	BUSYF
	Reset value																														0	0	1

Table 59. ICACHE 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
0x008	**ICACHE_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.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
	Reset value																														0	0	
0x00C	**ICACHE_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.	Res.	Res.	CERRF	CBSYENDF	Res.
	Reset value																														0	0	
0x010	**ICACHE_ HMONR**	HITMON[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	**ICACHE_ MMONR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018 to 0x01C	Reserved	Reserved																															
0x020	**ICACHE_CRR0**	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	Res.	BASEADDR[28:21]						
	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
0x024	**ICACHE_CRR1**	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	Res.	BASEADDR[28:21]						
	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
0x028	**ICACHE_CRR2**	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	Res.	BASEADDR[28:21]						
	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
0x02C	**ICACHE_CRR3**	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	Res.	BASEADDR[28:21]						
	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

Refer to *Section 2.3* for the register boundary addresses.

8 Power control (PWR)

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domains (VCORE)
  - VDD domain
  - Backup domain (VBAT)
  - Analog domain (VDDA)
  - Supply for the SMPS power stage (available on SMPS packages)
  - VDDIO2 domain on Port G
  - VDDUSB for USB transceiver
- System supply voltage regulation
  - SMPS step down converter
  - Voltage regulator (LDO)
  - External SMPS mode
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - PVM monitor (VDDA, VDDUSB, VDDIO2)
  - Temperature threshold monitor
  - Upper VDD voltage threshold monitor
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes
- VBAT battery charging
- TrustZone security

8.1 Power supplies and supply domains

The STM32L552xx and STM32L562xx devices require a 1.71 V to 3.6 V operating supply voltage (V_{DD}). Several peripherals are supplied through independent power domains: V_{DDA}, V_{DDIO2}, V_{DDUSB}. Those supplies must not be provided without a valid operating supply on the V_{DD} pin.

- V_{DD} = 1.71 V to 3.6 V

  V_{DD} is the external power supply for the I/Os, the internal regulator (or the SMPS step down converter depending on the device) and the system analog such as reset, power management and internal clocks. It is provided externally through VDD pins.
- V_{DDA} = 1.62 V (ADCs/COMPs) / 1.8 V (DACs/OPAMPs) / 2.4 V (VREFBUF) to 3.6 V

  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.

- VDDSMPS = 2 V to 3.6 V

  VDDSMPS is the external power supply for the SMPS step down converter. It is provided externally through VDDSMPS supply pin, and shall be connected to the same supply as VDD.

- V_{LXSMPS} is the switched SMPS step down converter output.
- V_{15SMPS} are the power supply for the system regulator. It is provided externally through the SMPS step down converter V_{LXSMPS} output.

Note: The SMPS power supply pins are available only on a specific package with SMPS step down converter option.

- VDD12 = 1.05 to 1.32 V

  VDD12 is the external power supply bypassing the internal regulator when connected to an external SMPS. It is provided externally through VDD12 pins and only available on packages with the external SMPS supply option. VDD12 does not require any external decoupling capacitance and cannot support any external load.

Note: The VDD12 power supply pins are available only on a specific package with external SMPS option.

- V_{DDUSB} = 3.0 V to 3.6 V

  V_{DDUSB} is the external independent power supply for USB transceivers. The V_{DDUSB} voltage level is independent from the V_{DD} voltage. V_{DDUSB} should be preferably connected to V_{DD} when the USB is not used.

  The V_{DDUSB} power supply may not be present as a dedicated pin, but to be internally bonded to V_{DD}. For such devices, V_{DD} has to respect the V_{DDUSB} supply range when the USB is used.

- V_{DDIO2} = 1.08 V to 3.6 V

  V_{DDIO2} is the external power supply for 14 I/Os (Port G[15:2]). The V_{DDIO2} voltage level is independent from the V_{DD} voltage and should preferably be connected to V_{DD} when PG[15:2] are not used.

- 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.

- V_{REF-}, V_{REF+}

  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 two 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.

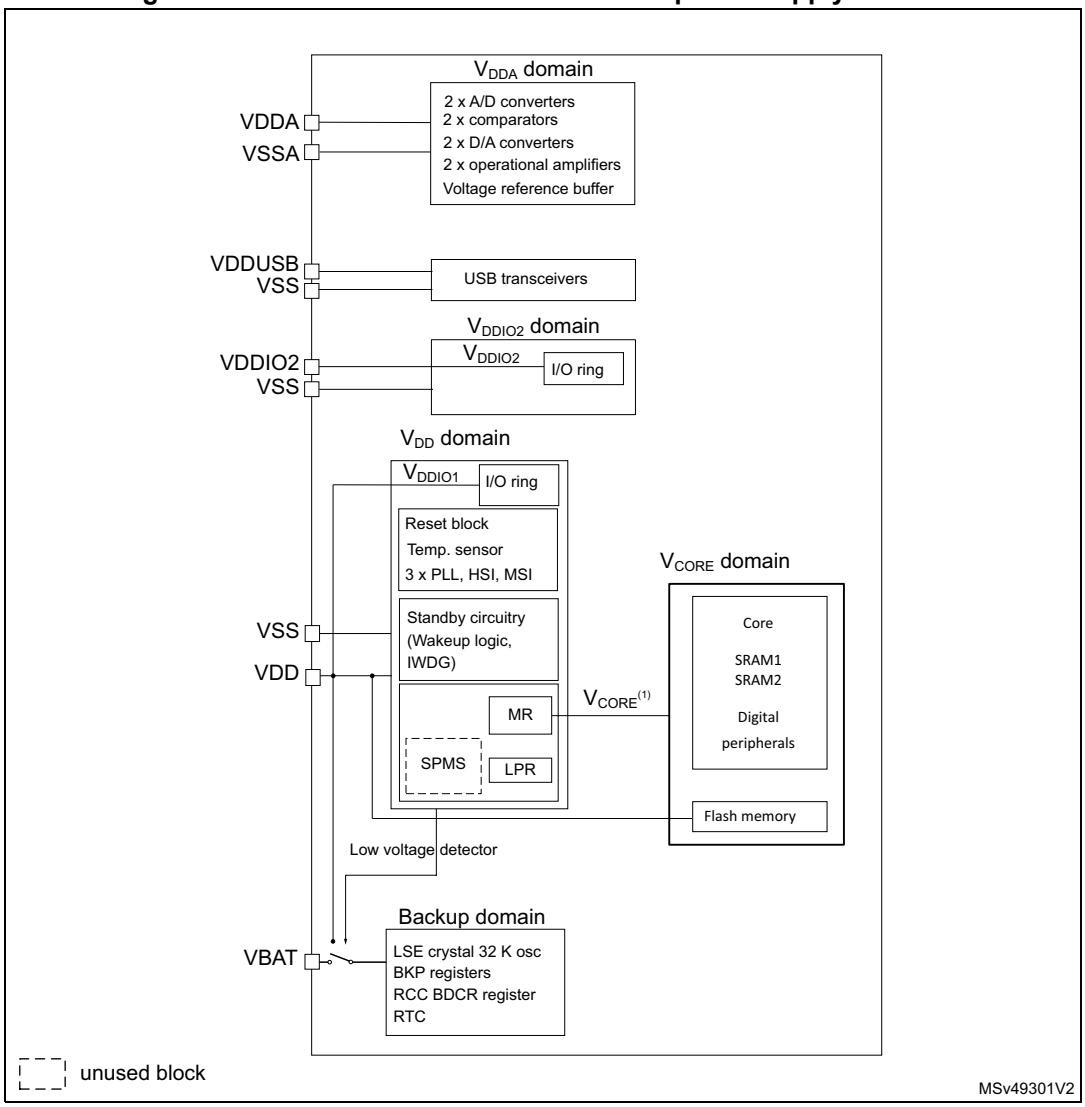
  On some packages, VREF- and VREF+ pins are not available. When not available on the package, they are internally bonded to respectively VSSA and VDDA.

When the VREF+ is double-bonded with VDDA in a package, the internal voltage reference buffer is not available and must be kept disable (refer to related device datasheet for packages pinout description).

V_{REF-} must always be equal to V_{SSA}.

In the STM32L552xx and STM32L562xx devices, the I/Os, the embedded LDO regulator and the system analog peripherals (such as PLLs and reset block) are fed by V_{DD} supply source. The embedded linear voltage regulator is used to supply the internal digital power V_{CORE}. V_{CORE} is the power supply for digital peripherals and memories.

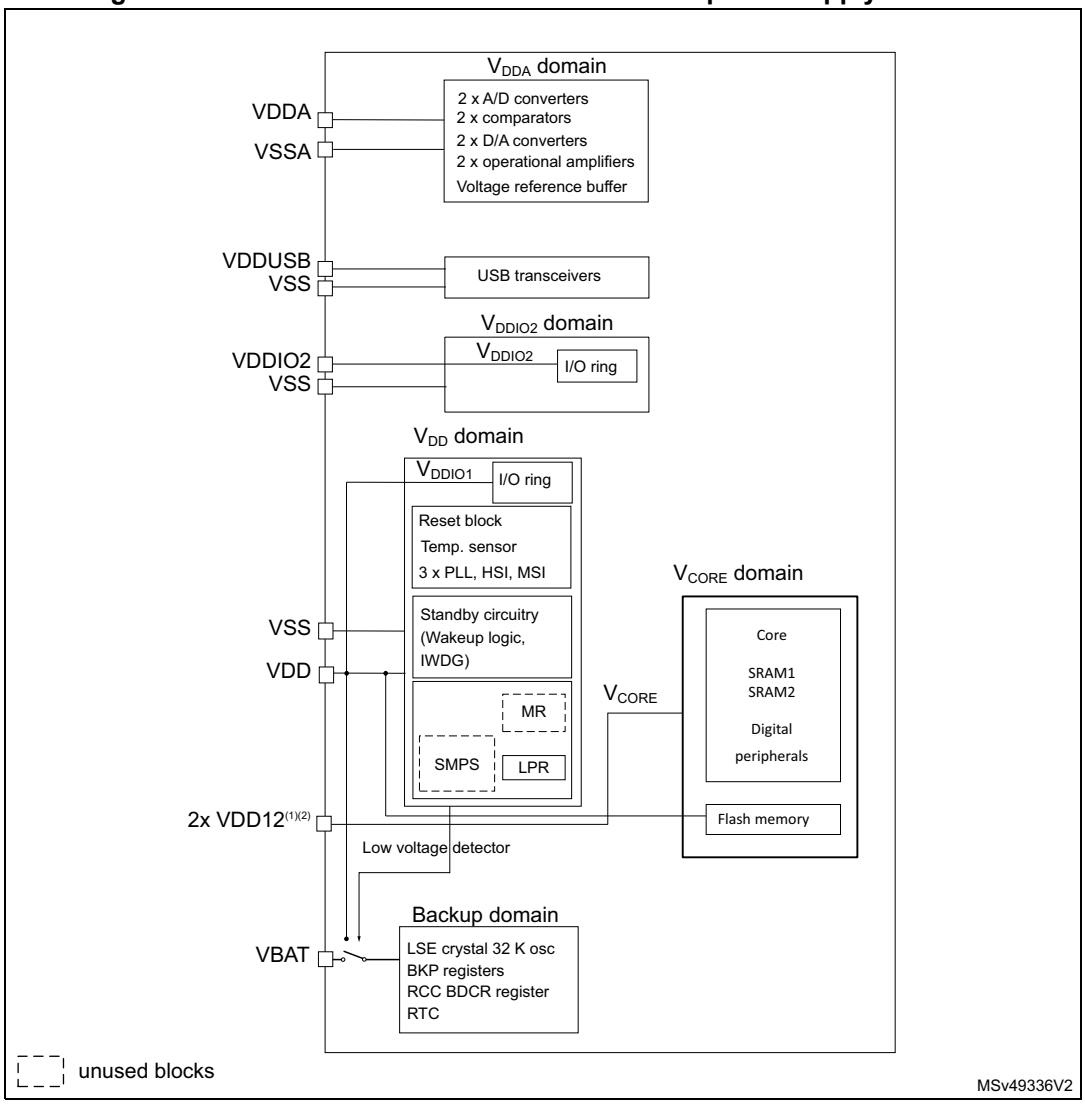
Figure 21. STM32L552xx and STM32L562xx power supply overview



1. V_{CORE} is provided by either MR or LPR, depending on the operating power mode.

In the STM32L552xxxP and STM32L562xxxP devices, the I/Os and system analog peripherals (such as PLLs, and reset block) are fed by V_{DD} supply source. The V_{CORE}, power supply for digital peripherals and memories is generated from external SMPS.

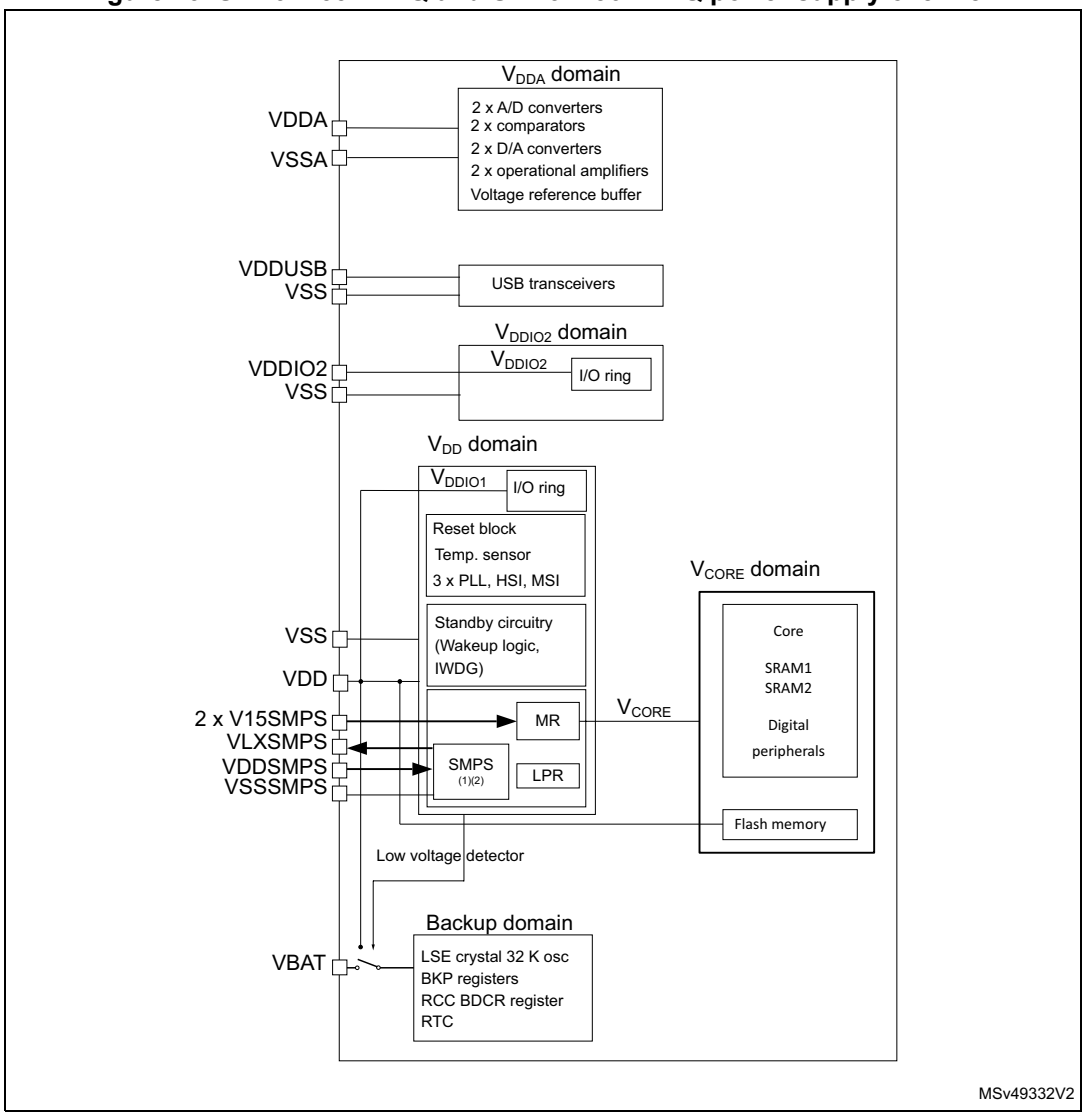
Figure 22. STM32L552xxxP and STM32L562xxxP power supply overview



1. If the selected package has the external SMPS option but no external SMPS is used by the application (the embedded LDO is used instead), the VDD12 pins are kept unconnected.
2. VDD12 is intended to be connected with external SMPS (switched-mode power supply) to generate the V_{CORE} logic supply in Run, Sleep and Stop 0 modes only.

In the STM32L552xxxQ and STM32L562xxxQ devices, the I/Os, the embedded SMPS step down converter and the system analog peripherals (such as PLLs and reset block) are fed by V_{DD} supply source. The embedded linear main voltage regulator that provides the V_{CORE} supply for digital peripherals and memories is fed by the SMSP step down converter output.

Figure 23. STM32L552xxxQ and STM32L562xxxQ power supply overview



1. Refer to *Figure 24* for SMPS step down converter power supply scheme.
2. During Low-power sleep, Low-power run, Stop 1, Stop 2, Standby and Shutdown modes, the SMPS step down converter is switched to Open mode. In Low-power sleep, Low-power run, Stop 1, Stop 2 and Standby with SRAM2 retention modes, the low-power regulator is used to provide the V_{CORE}. The SMPS is used in Run, Sleep and Stop 0 modes. It supplies the main regulator which provides the V_{CORE}.

Note: If the selected package has the SMPS step down converter option but the SMPS is not used by the application (and the embedded LDO is used instead), it is recommended to set the SMPS power supply pins as follows:
- VDDSMPS and VLXSMPS connected to VSS
- V15SMPSconnected to VDD.

8.1.1 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 (PVM), and compared with two thresholds (1.65 V for PVM3 or 1.8 V for PVM4), refer to *Section 8.3.3: 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 voltage reference is enabled by setting the ENVR bit in the *Section 23.4.1: VREFBUF control and status register (VREFBUF_CSR)*. The voltage reference is set to 2.5 V when the VRS bit is set and to 2.048 V when the VRS bit is cleared. The internal voltage reference can also provide the voltage to external components through V_{REF+} pin. Refer to the device datasheet and to *Section 23: Voltage reference buffer (VREFBUF)* for further information.

8.1.2 Independent I/O supply rail

Some I/Os from port G (PG[15:2]) are supplied from a separate supply rail. The power supply for this rail can range from 1.08 V to 3.6 V and is provided externally through the V_{DDIO2} pin. The V_{DDIO2} voltage level is completely independent from V_{DD} or V_{DDA}. The V_{DDIO2} pin is available only for some packages. Refer to the pinout diagrams or tables in the related device datasheet(s) for I/O list(s).

After reset, the I/Os supplied by V_{DDIO2} are logically and electrically isolated and therefore are not available. The isolation must be removed before using any I/O from PG[15:2], by setting the IOSV bit in the PWR_CR2 register, once the V_{DDIO2} supply is present.

The V_{DDIO2} supply is monitored by the peripheral voltage monitoring (PVM2) and compared with the internal reference voltage (3/4 V_{REFINT}, around 0.9V), refer to *Section 8.3.3: Peripheral voltage monitoring (PVM)* for more details.

8.1.3 Independent USB transceivers supply

The USB transceivers are supplied from a separate V_{DDUSB} power supply pin. V_{DDUSB} range is from 3.0 V to 3.6 V and is completely independent from V_{DD} or V_{DDA}.

After reset, the USB features supplied by V_{DDUSB} are logically and electrically isolated and therefore are not available. The isolation must be removed before using the USB peripheral, by setting the USV bit in the PWR_CR2 register, once the V_{DDUSB} supply is present.

The V_{DDUSB} supply is monitored by the peripheral voltage monitoring (PVM1) and compared with the internal reference voltage (V_{REFINT}, around 1.2 V), refer to *Section 8.3.3: Peripheral voltage monitoring (PVM)* for more details.

8.1.4 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 41.3: RTC functional description on page 1402*)
- PA0/TAMP_IN2/TAMP_OUT1 and PE6/TAMP_IN3/TAMP_OUT6 when they are configured by the tamper 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 (for example 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 41.3:*

RTC functional description)

- PA0/TAMP_IN2 and PE6/TAMP_IN3 when they are configured by the tamper 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 9.8.13: RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)*
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 *RCC Backup domain control register (RCC_BDCR)*.
4. Enable the RTC clock by setting the RTCEN [15] bit in the *RCC Backup domain control register (RCC_BDCR)*.

VBAT battery charging

When V_{DD} is present, It is possible to charge the external battery on VBAT through an internal resistance.

The VBAT charging is done either through a 5 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.

8.2 System supply voltage regulation

8.2.1 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 three different power ranges (Range 0, Range 1 and Range 2) in order to optimize the consumption depending on the system's maximum operating frequency (refer to *Section 9.3.10: Clock source frequency versus voltage scaling* and to *Section 6.3.3: Read access latency*).

At power-on reset or a system reset, the main regulator voltage Range 2 is selected by default.

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 and SRAM2.
- In Stop 1 and Stop 2 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, SRAM1and SRAM2.

- In Standby mode, the SRAM2 content can be fully or partially (only 4 Kbytes) preserved (depending on RRS[1:0] bits 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) and SRAM1 are powered off.
- In Standby mode, both regulators are powered off. The contents of the registers, SRAM1 and SRAM2 is lost except for the Standby circuitry and the backup domain.
- In Shutdown mode, both regulators are powered off. When exiting from Shutdown mode, a power-on reset is generated. Consequently, the contents of the registers, all SRAMs is lost, except for the backup domain.

8.2.2 Embedded SMPS step down converter

The built-in SMPS step down converter is a power-efficient DC/DC non-linear switching regulator that improves low-power performance when the VDD voltage is high enough. The SMPS step down converter automatically enters in Bypass mode when the V_{DD} voltage falls below a VDD minimum value following the selected voltage range and switched back to the selected operating mode when V_{DD} rises above the minimum value.

When the SMPS step down converter is enabled, it can be configured in:

- High-power mode (HPM): achieving a high efficiency at high current load.

  SMPS high-power mode is used in all ranges (0, 1 and 2).

  It is the default selected mode after POR reset.
- Low-power mode (LPM): achieving a high efficiency at low current load.

  When enabled, the voltage scaling must not be modified. This mode shall be only selected in Range 2 and when power consumption does not exceed 30 mA.
- Bypass mode:

  When the Bypass mode is enabled, the SMPS step down converter is switched OFF and it is possible to change the voltage scaling. This mode can be forced by software by setting the SMPSBYP bit in PWR_CR4 register. The Bypass mode can be enabled or disabled on the fly at any time by the application software whatever the selected operation mode.

  In Range 0 and Range 1, the SMPS Bypass mode is selected automatically when V_{DD} drops below 2.05 V.

  In Range 2, if V_{DD} is less than 2.05 V, the SMPS Bypass mode must be forced by software. For this purpose, the PVD0 should monitor the V_{DD} power supply and the software must force bypass mode by setting the SMPSBYP bit in PWR_CR4 register.

  The following table summarizes the SMPS behavior depending on the main regulator range, V_{DD} and consumption.

Table 60. SMPS modes summary

Ranges	Max AHB clock	V_{CORE}	SMPS mode	
			$V_{DD} \leq 2.05$ V	$V_{DD} > 2.05$ V
Range 0	110 MHz	1.28 V	Automatic Bypass mode $V_{15SMPS} = V_{DD}$	HP mode Max current consumption = 120 mA V_{15SMPS} = 1.6 V
Range 1	80 MHz	1.2 V	Automatic Bypass mode $V_{15SMPS} = V_{DD}$	HP mode Max current consumption = 80 mA V_{15SMPS} = 1.5 V
Range 2	26 MHz	1.0 V	Software Bypass mode(1) $V_{15SMPS} = V_{DD}$	LP mode or HP mode Max current consumption = 30 mA V_{15SMPS} = 1.3 V

1. There is no automatic SMPS bypass in Range 2. The user application should use PVD0 to monitor V_{DD} supply and request the SMPS Bypass mode.

Refer to *Table 61* for SMPS step down converter operating mode.

Table 61. SMPS step down converter operating mode

SMPSBYP bit	SMPSLPEN bit	SMPSHPRDY flag	Description
0	0	1	High-power mode
0	1	0	Low-power mode
1	x	x	Bypass mode

The sequence to go from Low-power to High-power mode is:

1. Clearing the SMPSLPEN bit in the PWR_CR4 register
2. Check that the SMPSHPRDY flag is set in PWR_SR1 register

The sequence to go from High-power to Low-power mode is:

1. Configure the system frequency and voltage scaling
2. Set the SMPSLPEN bit in the PWR_CR4 register

Note: If the Bypass mode is active and the SMPSLPEN bit is set, the switch to SMPS Low-power mode is delayed until the SMPS step down converter exits the Bypass mode.

Note: The SMPS step down converter operating mode and Voltage scaling range selection shall be changed only in RUN mode.

8.2.3 SMPS step down converter power supply scheme

The SMPS step down converter requires an external coil with typical value of 4.7 µH to be connected between the dedicated V_{LXSMPS} pin to V_{SSSMPS} via a capacitor of 4.7µF. It is switched OFF when:

- LPREG is used

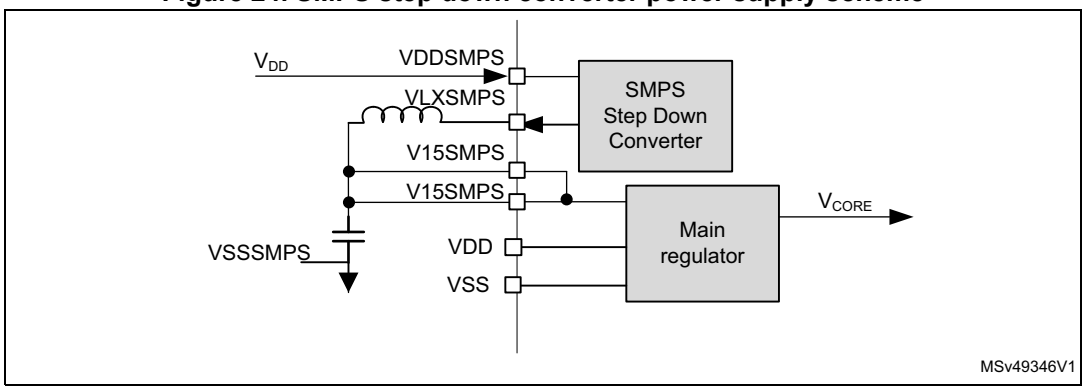
or

- SMPS is configured in Bypass mode by software in RUN/SLEEP modes (SMPSBYP bit set in PWR_CR4).

Thus, only main regulator is used by the application.

Refer to *Figure 24* below.

Figure 24. SMPS step down converter power supply scheme



If the selected package is with the SMPS step down converter option but it is never used by the application, it is recommended to set the SMPS power supply pins as follows:

- V_{DDSMPS} and V_{LXSMPS} connected to VSS
- V_{15SMPS}connected to VDD

8.2.4 SMPS step down converter versus low-power mode

The SMPS step down converter operating mode depends on the selected operating and upon the system operating modes (LP)Run, (LP)Sleep, Stop 0, Stop 1, Stop 2, Standby, and Shutdown.

During Stop 1, Stop 2, Standby and Shutdown modes the SMPS step down converter is switched to Open mode (see *Table 62*). When exiting from low-power modes (except Shutdown) the SMPS step down converter is set by hardware to the mode selected prior to the low-power mode selection. The SMPS step down converter mode configuration bits in PWR_CR4 (SMPSLEN or SMPSBYP) are retained in Standby mode.

After POR reset, the SMPS step down converter is in High-power mode.

Table 62. SMPS step down converter versus low-power modes

System operating mode	SMPS step down converter state	Description
Run, Sleep	ON	SMPS in HPM or LPM mode and, in ranges 0/1 it switches to Bypass mode following VDD minimum value versus the selected voltage range
Stop 0	ON	SMPS in HPM or LPM mode
Stop 1, Stop 2, Low power run, Low power sleep, Standby with SRAM2 retention	Open	SMPS is bypassed, LPR regulator is used
Standby and Shutdown	Open	SMPS is bypassed

Note: It is recommended to enable the SMPS bypass mode prior entering Stop modes in order to reduce the wake up time.

Note: If the Bypass mode is requested, entering Stop 1 or Stop 2 or Low-power run or Low-power sleep modes shall be delayed until the SMPS BYPASS ready flag SMPSBYPRDY is set.

Note: In Low-power run mode, the following bits shall not be modified (PWR_CR4: SMPSLPEN, SMPSFSTEN, SMPSBYP).

SMPS step down converter fast startup

After POR reset, the SMPS step down converter starts in High-power mode and in Slow-startup mode. The low-startup feature is selected to limit the inrush current after power-on reset.

However, it is possible to configure a faster startup on the fly and it is applied for next SMPS startup (after a wakeup from low-power mode - except Shutdown and VBAT modes) or for next SMPS output voltage change (Bypass or VOS change). The fast startup is selected by setting the SMPSFSTEN bit in the PWR_CR4 register.

Note: Setting the SMPSFSTEN bit in the PWR_CR4 register allows also a faster switch to Bypass mode (i.e. V_{15SMPS} reaching V_{DD}).

Note: The timing needed for Bypass mode to be effective (i.e. SMPSBYPRDY flag is set) is counted starting from V15 already at target value (1.3V (Range 2); 1.5V (Range 1); 1.6V (Range 0)). This timing is determined by the parameter V15 slew rate, depending itself on the fact that fast startup is enabled or disabled. Refer to the device datasheet.

8.2.5 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 the device to improve its 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.

The main regulator operates in the following ranges:

- Main regulator Range 0: high performance;

  It provides a typical output voltage at 1.28 V. It is used when the system clock frequency is up to 110 MHz. The Flash access time for read access is minimum, write and erase operations are possible.

- Main regulator Range 1: medium performance;

  It provides a typical output voltage at 1.2 V. It is used when the system clock frequency is up to 80 MHz. The Flash access time for read access is minimum, write and erase operations are possible.

- Main regulator Range 2: low-power range.

  It 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[1:0] bits in the PWR_CR1 register. The main regulator voltage Range 2 is selected by default.

Note: In Low-power run mode, the VOS[1:0] must not be modified.

Voltage scaling and SMPS step down converter

When the SMPS step down converter is selected, the VOS[1:0] bits must not be modified when the SMPS is in a low-power mode.

When the voltage scaling is updated, the SMPS step down converter output voltage is scaled automatically following the selected voltage range.

The sequence to go from Range 0 /Range 1 to Range 2 is:

1. In case of switching from Range 0 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 1 us 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[1:0] bits to “10” in the PWR_CR1 register.

The sequence to go from Range 2 to Range 1/Range 0 is:

1. Program the VOS[1:0] 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 0 or 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 <= 80 MHz:

  - Configure and switch to PLL for a new system frequency.

– If the system frequency is SYSCLK > 80 MHz:

  - The system clock must be divided by 2 using the AHB prescaler before switching to a higher system frequency.

  - 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 Range 1 to Range 0 is:

1. The system clock must be divided by 2 using the AHB prescaler before switching to a higher system frequency.
2. Adjust the number of wait states according to the new frequency target in range1
3. Configure and switch to new system frequency.
4. Wait for at least 1us and then reconfigure the AHB prescaler to get the needed HCLK clock frequency.

The sequence to switch from Range 0 boost mode to Range 1 mode is:

1. Adjust the number of wait states according to the new frequency target in Range 0 default mode
2. Configure and switch to new system frequency.

8.2.6 VDD12 domain and external SMPS

VDD12 is intended to be connected with external SMPS (switched-mode power supply) to generate the V_{CORE} logic supply in Run, Sleep and Stop 0 modes only.

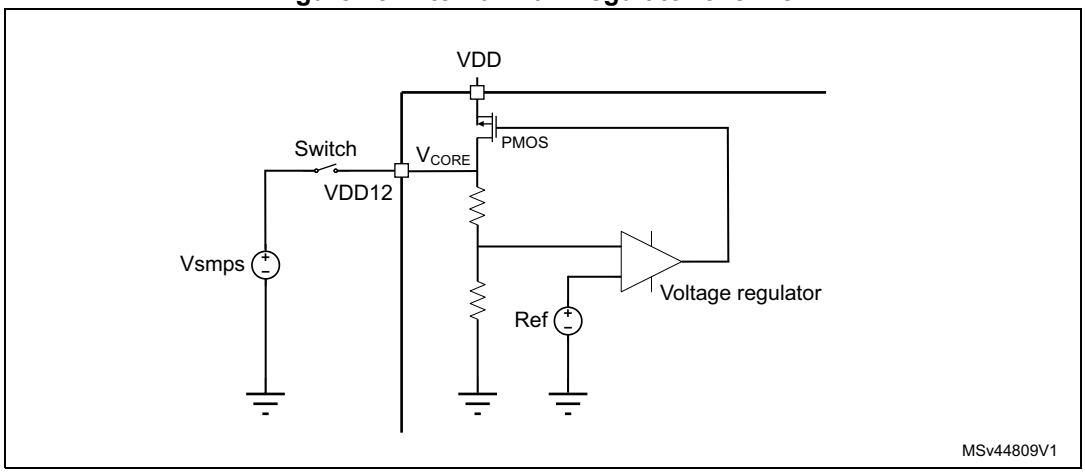
VDD12 pins correspond to the internal V_{CORE} powering the digital part of Core, RAMs, FLASH and peripherals. This significantly improves the power consumption with a gain from 50% or more depending of the external SMPS performances.

The main benefit occurs in Run and Sleep modes whereas in Stop 0 mode, the gain is less significant.

The figure below shows a schematic to understand how the internal regulator stops supplying V_{CORE} when an external voltage VDD12 is provided.

As VDD12 shares the same pin as output of the internal regulator, applying a slightly higher voltage (typically +50 mV) on the VDD12 blocks, the PMOS and the regulator consumption is negligible.

Figure 25. Internal main regulator overview



A switch, controlled by the chosen GPIO, is inserted between the external SMPS output and VDD12.

There are two possible states:

- Connected: Switch is closed so external SMPS powers VDD12
- Disconnected: Switch is open and VDD12 is disconnected from external SMPS output

Proper software management through GPIOs to enable/disable external SMPS and to connect/disconnect external SMPS through the switch, is required to conform with the rules described below. See also *Section 8.2.5: Dynamic voltage scaling management*.

It is mandatory to respect the following rules to avoid any damage or instability on either digital parts or internal regulators:

- In Run, Sleep and Stop 0 modes, VDD12 can be connected and should respect
  - VDD12 < 1.32 V
  - VDD12 ≥ V_{CORE} + 50 mV giving for main regulator

    Range 0, V_{CORE} =1.28 V so VDD12 should be greater than 1.33 V, but this cannot match previous rule VDD12 < 1.32 V, so it is not a functional use case with an external SMPS.

    Range 1, V_{CORE} = 1.2 V so VDD12 should be greater than 1.25 V.

    Range 2, V_{CORE} = 1.0 V so VDD12 should be greater than 1.05 V
  - VDD12 ≥ 1.08 V in Range 1 when 80 MHz ≥ SYSCLK frequency ≥ 26 MHz
  - VDD12 ≥ 1.14 V in Range 0 when SYSCLK frequency > 80 MHz
- In all other modes, such as LPRun, LPSleep, Stop 1, Stop 2, Standby and Shutdown modes, VDD12 must be disconnected from external SMPS output. This means that the pin must be connected to a high impedance output:
  - VDD12 connected to HiZ (voltage is provided by internal regulators)
- Transitions of VDD12 from connected to disconnected is only allowed when SYSCLK frequency ≤ 26 MHz to avoid to big voltage drop on main regulator side.

Note: In case of asynchronous reset while having the V_{DD12} ≤ 1.25 V, VDD12 should switch to HiZ in less than regulator switching time from Range 2 to Range 1 (~1 us).

Note: V_{DD12} Range 2 is extended down to 1.00 V for better efficiency, thus following formula applies when bit EXTSMPSEN in the Power control register 4 (PWR_CR4) is set: Range 2, VCORE = 0.95 V so VDD12 should be greater than 1.00 V

Note: For more details on V_{DD12} management, refer to AN4978 "Design recommendations for STM32L4xxxx with external SMPS, for ultra-low-power applications with high performance".

8.3 Power supply supervision

8.3.1 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.

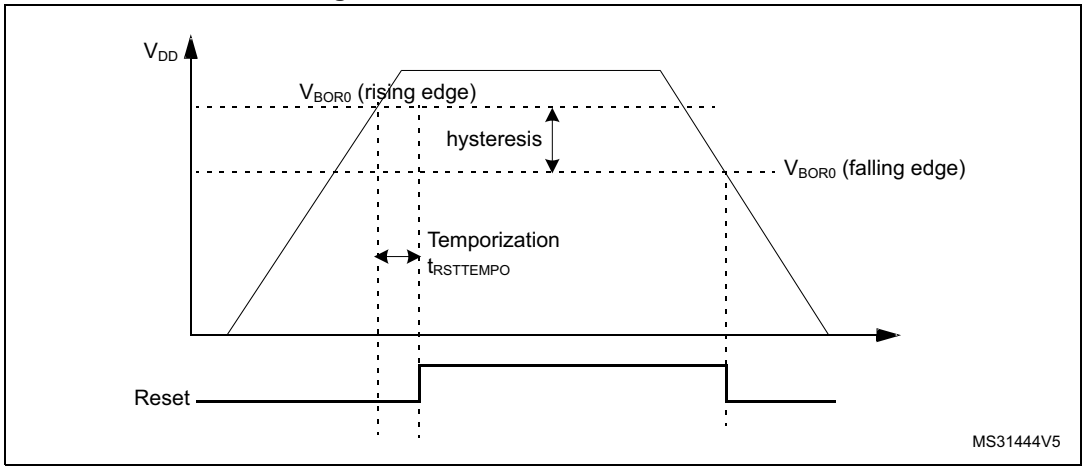
During power-on, the BOR keeps the device under reset until the supply voltage V_{DD} reaches the specified V_{BORx} threshold. When V_{DD} drops below the selected threshold, a device reset is generated. When V_{DD} is above the V_{BORx} upper limit, the device reset is released and the system can start.

For more details on the brown-out reset thresholds, refer to the electrical characteristics section in the datasheet.

During Stop 2 and Standby modes, it is possible to set the BOR in Ultra-low-power mode to further reduce the current consumption by setting the ULPMEN bit in PWR_CR3 register.

For Stop 2 mode, the BOR Ultra-low-power mode can be set if the BORH is set, otherwise there is no power consumption optimization.

Figure 26. Brown-out reset waveform



1. The reset temporization $t_{RSTTEMPO}$ is present only for the BOR lowest threshold (V_{BOR0}).

8.3.2 Programmable voltage detector (PVD)

The user 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.

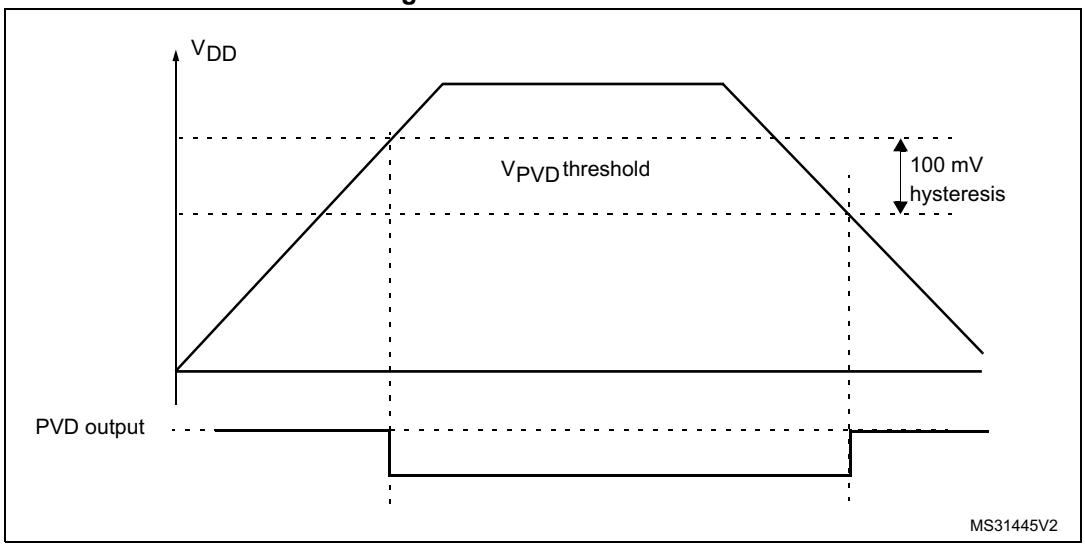
A PVDO flag is available, in the *Power control register 2 (PWR_CR2)*, to indicate if V_{DD} is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers.

The rising/falling edge sensitivity of the EXTI Line16 should be configured according to PVD output behavior. For example, if the EXTI line 16 is configured to rising edge sensitivity, the interrupt is generated when VDD drops below the PVD threshold. As an example the service routine could perform emergency shutdown tasks.

During during Stop 2 and Standby modes, it is possible to set the PDV in Ultra-low-power mode to further reduce the current consumption by setting the ULPMEN bit in PWR_CR3 register.

For Stop 2 mode, the PVD Ultra-low-power mode can be set if the PVD is set, otherwise if PVD is disabled, there is no power consumption optimization.

Figure 27. PVD thresholds



8.3.3 Peripheral voltage monitoring (PVM)

Only V_{DD} is monitored by default, as it is the only supply required for all system-related functions. The other supplies (V_{DDA}, V_{DDIO2} and V_{DDUSB}) can be independent from V_{DD} and can be monitored with four peripheral voltage monitoring (PVM).

Each of the four PVMx (x=1, 2, 3, 4) is a comparator between a fixed threshold V_{PVMx} and the selected power supply. PVMOx flags indicate if the independent power supply is higher or lower than the PVMx threshold: PVMOx flag is cleared when the supply voltage is above the PVMx threshold, and is set when the supply voltage is below the PVMx threshold.

Each PVM output is connected to an EXTI line and can generate an interrupt if enabled through the EXTI registers. The PVMx output interrupt is generated when the independent power supply drops below the PVMx threshold and/or when it rises above the PVMx threshold, depending on EXTI line rising/falling edge configuration.

Each PVM can remain active in Stop 0, Stop 1 and Stop 2 modes, and the PVM interrupt can wake up from the Stop mode.

Table 63. PVM features

PVM	Power supply	PVM threshold	EXTI line
PVM1	V_{DDUSB}	V_{PVM1} (around 1.2 V)	35
PVM2	V_{DDIO2}	V_{PVM2} (around 0.9 V)	36
PVM3	V_{DDA}	V_{PVM3} (around 1.65 V)	37
PVM4	V_{DDA}	V_{PVM4} (around 1.8 V)	38

The independent supplies (V_{DDA}, V_{DDIO2} and V_{DDUSB}) are not considered as present by default, and a logical and electrical isolation is applied to ignore any information coming from the peripherals supplied by these dedicated supplies.

- If these supplies are shorted externally to V_{DD}, the application should assume they are available without enabling any peripheral voltage monitoring.
- If these supplies are independent from V_{DD}, the peripheral voltage monitoring (PVM)

can be enabled to confirm whether the supply is present or not.

The following sequence must be done before using the USB peripheral:

1. If V_{DDUSB} is independent from V_{DD}:
   a) Enable the PVM1 by setting PVME1 bit in the *Power control register 2 (PWR_CR2)*.
   b) Wait for the PVM1 wakeup time.
   c) Wait until PVMO1 bit is cleared in the *Power status register 2 (PWR_SR2)*.
   d) Optional: Disable the PVM1 for consumption saving.
2. Set the USV bit in the *Power control register 2 (PWR_CR2)* to remove the V_{DDUSB} power isolation.

The following sequence must be done before using any I/O from PG[15:2]:

1. If V_{DDIO2} is independent from V_{DD}:
   a) Enable the PVM2 by setting PVME2 bit in the *Power control register 2 (PWR_CR2)*.
   b) Wait for the PVM2 wakeup time.
   c) Wait until PVMO2 bit is cleared in the *Power control register 2 (PWR_CR2)*.
   d) Optional: Disable the PVM2 for consumption saving.
2. Set the IOSV bit in the *Power control register 2 (PWR_CR2)* to remove the V_{DDIO2} power isolation.

The following sequence must be done before using any of these analog peripherals: analog to digital converters, digital to analog converters, comparators, operational amplifiers, voltage reference buffer:

1. If V_{DDA} is independent from V_{DD}:
   a) Enable the PVM3 (or PVM4) by setting PVME3 (or PVME4) bit in the *Power control register 2 (PWR_CR2)*.
   b) Wait for the PVM3 (or PVM4) wakeup time.
   c) Wait until PVMO3 (or PVMO4) bit is cleared in the *Power status register 2 (PWR_SR2)*.
   d) Optional: Disable the PVM3 (or PVM4) for consumption saving.

Enable the analog peripheral, which automatically removes the V_{DDA} isolation.

8.3.4 Upper voltage threshold monitoring

The upper VDD voltage monitoring is enabled by setting the bit VMONEN in the TAMP_CFGR register.

If the upper VDD voltage monitoring internal tamper is enabled in the TAMP peripheral (ITAMP1E=1 in the TAMP_CR1 register), a tamper event is generated when the VDD domain voltage is above the specified threshold.

The upper VDD monitoring can be periodical. This feature is enabled by setting the bit WUTMONEN in the TAMP configuration register TAMP_CFGR.

In this case, the monitoring is controlled by the RTC wakeup timer PWM resulting from the WUTF flag automatic clear and depending on the bitsfield WUTOCLR in the RTC_WUTR register. For more details, refer to the RTC section.

The monitoring is enabled during the PWM high level and disabled during the PWM low level.

Note: For threshold value, refer to the product datasheet.

Note: In case the VDD is below the functional range, a Brown-out reset is generated.

8.3.5 Temperature threshold monitoring

The temperature monitoring is enabled by setting the bit TMONEN in the TAMP_CFGR register.

If the temperature monitoring internal tamper is enabled in the TAMP peripheral (ITAMP2E=1 in the TAMP_CR1 register), a tamper event is generated when the temperature is above or below the specified thresholds.

The temperature monitoring can be periodical. This feature is enabled by setting the bit WUTMONEN in the TAMP configuration register TAMP_CFGR.

In this case, the monitoring is controlled by the RTC wakeup timer PWM resulting from the WUTF flag automatic clear and depending on the bitfield WUTOCLR in the RTC_WUTR register. For more details, refer to the RTC section.

The monitoring is enabled during the PWM high level and disabled during the PWM low level.

Note: For thresholds values, refer to the product datasheet.

8.4 Power management

8.4.1 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 eight low-power modes:

- Sleep mode: CPU clock off, all peripherals including Cortex®-M33 core such as NVIC and SysTick can run and wake up the CPU when an interrupt or an event occurs. Refer to *Section 8.4.5: Sleep mode*.
- Low-power run mode: This mode is achieved when the system 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 8.4.3: Low-power run mode (LP run)*.
- Low-power sleep mode: This mode is entered from the Low-power run mode: Cortex®-M33 is off. Refer to *Section 8.4.6: Low-power sleep mode (LP sleep)*.
- Stop 0, Stop 1 and Stop 2 modes: SRAM1, SRAM2 and all registers content are retained. All clocks in the V_{CORE} domain are stopped, the PLL, the MSI, the HSI16 and the HSE are disabled. The LSI and the LSE can be kept running.

  The RTC 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 2 mode, most of the V_{CORE} domain is put in a lower leakage mode.

Stop 1 offers the largest number of active peripherals and wakeup sources, a smaller wakeup time but a higher consumption than Stop 2. In Stop 0 mode, the main regulator remains ON, which allows the fastest wakeup time but with much higher consumption. The active peripherals and wakeup sources are the same as in Stop 1 mode.

The system clock, when exiting from Stop 0, Stop 1 or Stop 2 mode, can be either MSI up to 48 MHz or HSI16, depending on the software configuration.

Refer to *Section 8.4.7: Stop 0 mode* and *Section 8.4.9: Stop 2 mode*.

- Standby mode: V_{CORE} domain is powered off. However, it is possible to preserve the full SRAM2 content or only 4 Kbytes:
  - Standby mode with full or only the upper 4 Kbytes of SRAM2 retention when the RRS[1:0] bits are set to ‘01’ or ‘10’ respectively in the PWR_CR3 register. In this case, the SRAM2 is supplied by the low-power regulator.
  - Standby mode when the RRS[1:0] bits are 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 MSI, the HSI16 and the HSE 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 MSI from 1 MHz up to 8 MHz.

  Refer to *Section 8.4.10: Standby mode*.
- Shutdown mode: V_{CORE} domain is powered off. All clocks in the V_{CORE} domain are stopped, the PLL, the MSI, the HSI16, the LSI and the HSE are disabled. The LSE can be kept running. The system clock, when exiting the Shutdown mode, is MSI at 4 MHz. In this mode, the supply voltage monitoring is disabled and the product behavior is not guaranteed in case of a power voltage drop. Refer to *Section 8.4.11: 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 28. Low-power modes possible transitions

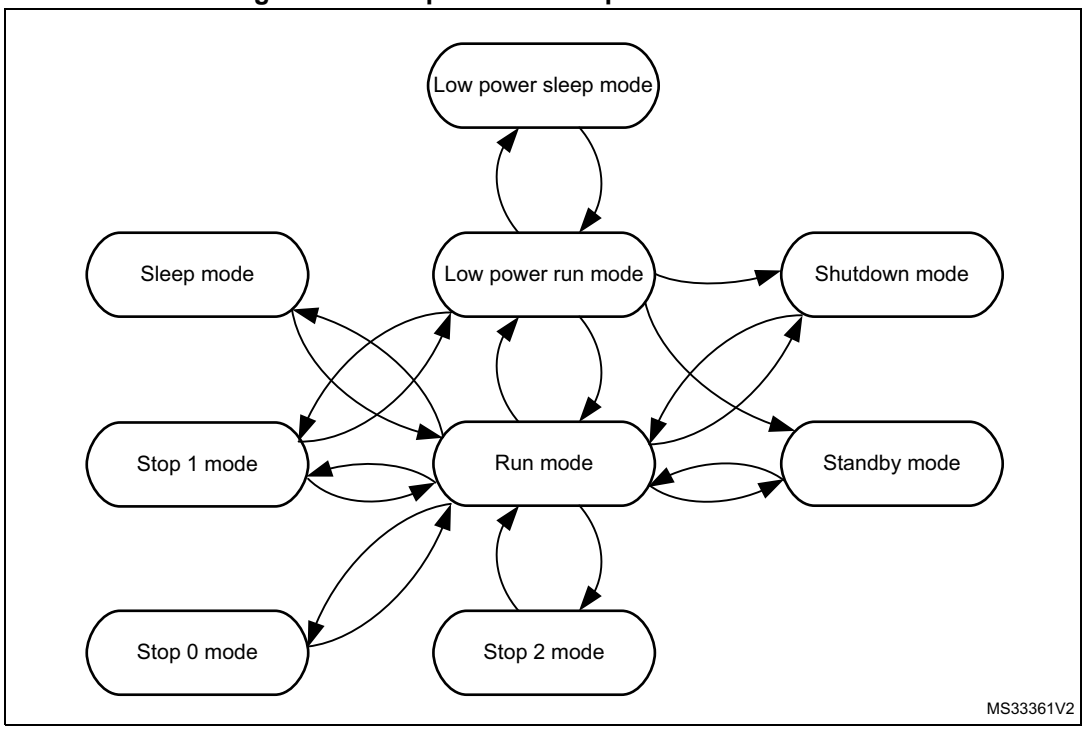


Table 64. Low-power mode summary

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	ON
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	OFF	ON
	Set LPR bit + WFE	Wakeup event			OFF	ON
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 when STOPWUCK=1 in RCC_CFGR MSI with the frequency before entering the Stop mode when STOPWUCK=0.	All clocks OFF except LSI and LSE	ON	ON
Stop 1	LPMS="001" + SLEEPDEEP bit + WFI or Return from ISR or WFE				OFF	
Stop 2	LPMS="010" + SLEEPDEEP bit + WFI or Return from ISR or WFE					
Standby with SRAM2_4KB	LPMS="011"+ Set RRS[1:0] bits to "10"+ SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, external reset in NRST pin, IWDG reset	MSI from 1 MHz up to 8 MHz			
Standby with SRAM2_Full	LPMS="011"+ Set RRS bits to "01"+ SLEEPDEEP bit + WFI or Return from ISR or WFE					
Standby	LPMS="011" + Clear RRS bits + SLEEPDEEP bit + WFI or Return from ISR or WFE				OFF	OFF
Shutdown	LPMS="1--" + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, external reset in NRST pin	MSI 4 MHz	All clocks OFF except LSE	OFF	OFF

1. Refer to *Table 65: Functionalities depending on the working mode*.

Table 65. Functionalities depending on the working mode(1)

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Stop 2		Standby		Shutdown		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
CPU	Y	-	Y	-	-	-	-	-	-	-	-	-	-
Flash memory (2 Mbytes)	O(2)	O(2)	O(2)	O(2)	-	-	-	-	-	-	-	-	-
SRAM1 (192 Kbytes)	Y	Y(3)	Y	Y(3)	Y	-	Y	-	-	-	-	-	-
SRAM2 (64 Kbytes)	Y	Y(3)	Y	Y(3)	Y	-	Y	-	O(4)	-	-	-	-
FSMC	O	O	O	O	-	-	-	-	-	-	-	-	-
OctoSPI	O	O	O	O	-	-	-	-	-	-	-	-	-
OTFDEC	O	O	O	O	-	-	-	-	-	-	-	-	-
Backup registers	Y	Y	Y	Y	Y	-	Y	-	Y	-	Y	-	Y
Brownout reset (BOR)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-
Programmable voltage detector (PVD)	O	O	O	O	O	O	O	O	-	-	-	-	-
Peripheral voltage monitor (PVMx; x=1,2,3,4)	O	O	O	O	O	O	O	O	-	-	-	-	-
DMA	O	O	O	O	-	-	-	-	-	-	-	-	-
High speed internal (HSI16)	O	O	O	O	(5)	-	(5)	-	-	-	-	-	-
Oscillator HSI48	O	O	-	-	-	-	-	-	-	-	-	-	-
High speed external (HSE)	O	O	O	O	-	-	-	-	-	-	-	-	-
Low speed internal (LSI)	O	O	O	O	O	-	O	-	O	-	-	-	-
Low speed external (LSE)	O	O	O	O	O	-	O	-	O	-	O	-	O
Multi speed internal (MSI)	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	O	O	-	-	-

Table 65. Functionalities depending on the working mode[1] (continued)

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Stop 2		Standby		Shutdown		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
V_{DD} voltage monitoring, temperature monitoring	O	O	O	O	O	O	O	O	O	O	-	-	-
RTC / TAMP	O	O	O	O	O	O	O	O	O	O	O	O	O
Number of RTC Tamper pins	8	8	8	8	8	O	8	O	8	O	8	O	3
USB, UCPD	O[8]	O[8]	-	-	-	O	-	-	-	-	-	-	-
USARTx (x=1,2,3,4,5)	O	O	O	O	O[6]	O[6]	-	-	-	-	-	-	-
Low-power UART (LPUART)	O	O	O	O	O[6]	O[6]	O[6]	O[6]	-	-	-	-	-
I2Cx (x=1,2,4)	O	O	O	O	O[7]	O[7]	-	-	-	-	-	-	-
I2C3	O	O	O	O	O[7]	O[7]	O[7]	O[7]	-	-	-	-	-
SPIx (x=1,2,3)	O	O	O	O	-	-	-	-	-	-	-	-	-
FDCAN1	O	O	O	O	-	-	-	-	-	-	-	-	-
SDMMC1	O	O	O	O	-	-	-	-	-	-	-	-	-
SAIx (x=1,2)	O	O	O	O	-	-	-	-	-	-	-	-	-
DFSDM1	O	O	O	O	-	-	-	-	-	-	-	-	-
ADCx (x=1,2)	O	O	O	O	-	-	-	-	-	-	-	-	-
DACx (x=1,2)	O	O	O	O	O	-	-	-	-	-	-	-	-
VREFBUF	O	O	O	O	O	-	-	-	-	-	-	-	-
OPAMPx (x=1,2)	O	O	O	O	O	-	-	-	-	-	-	-	-
COMPx (x=1,2)	O	O	O	O	O	O	O	O	-	-	-	-	-
Temperature sensor	O	O	O	O	-	-	-	-	-	-	-	-	-
Timers (TIMx)	O	O	O	O	-	-	-	-	-	-	-	-	-
Low-power timer 1, 3 (LPTIM1 and LPTIM3)	O	O	O	O	O	O	O	O	-	-	-	-	-
Low-power timer 2 (LPTIM2)	O	O	O	O	O	O	-	-	-	-	-	-	-

Table 65. Functionalities depending on the working mode[1] (continued)

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Stop 2		Standby		Shutdown		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
Independent watchdog (IWDG)	O	O	O	O	O	O	O	O	O	O	-	-	-
Window watchdog (WWDG)	O	O	O	O	-	-	-	-	-	-	-	-	-
SysTick timer	O	O	O	O	-	-	-	-	-	-	-	-	-
Touch sensing controller (TSC)	O	O	O	O	-	-	-	-	-	-	-	-	-
Random number generator (RNG)	O[8]	O[8]	-	-	-	-	-	-	-	-	-	-	-
AES hardware accelerator	O	O	O	O	-	-	-	-	-	-	-	-	-
HASH hardware accelerator	O	O	O	O	-	-	-	-	-	-	-	-	-
PKA	O	O	O	O	-	-	-	-	-	-	-	-	-
CRC calculation unit	O	O	O	O	-	-	-	-	-	-	-	-	-
GPIOs	O	O	O	O	O	O	O	O	[9]	5 pins [10]	[11]	5 pins [10]	-

1. Legend: Y = yes (enable). O = optional (disable by default, can be enabled by software). - = not available.

Gray cells highlight the wakeup capability in each mode.

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. 4 Kbytes or full SRAM2 content is preserved depending on RRS[1:0] bits configuration 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, Stop 1, Stop 2, Standby or Shutdown mode while the debug features are used. This is due to the fact that the Cortex®-M33 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 52.2.5: Debug and low-power modes*.

8.4.2 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 9.8.3: RCC 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.

8.4.3 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 system frequency should not exceed 2 MHz.

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 system 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 66: 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 system clock frequency.

Refer to *Table 66: Low-power run* on how to exit the Low-power run mode.

Table 66. Low-power run

Low-power run mode	Description
Mode entry	Decrease the system clock frequency below 2 MHz LPR = 1
Mode exit	LPR = 0 Wait until REGLPF = 0 Increase the system clock frequency
Wakeup latency	Regulator wakeup time from low-power mode

8.4.4 Low-power modes

Entering into a 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®-M33 system control register is set on *Return from ISR*.

Entering into a low-power mode through WFI or WFE is executed only if no interrupt is pending or no event is pending.

Exiting from a low-power mode

From Sleep mode and Stop mode the MCU exits the 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®-M33 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®-M33 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 wakes up the MCU, even the disabled ones. Only enabled NVIC interrupts with sufficient priority wake up 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 exits the 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 395: RTC block diagram*).
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.).

8.4.5 Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

Entering the Sleep mode

The Sleep mode is entered according *Section : Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex®-M33 System Control register is clear.

Refer to *Table 67: Sleep mode* for details on how to enter the Sleep mode.

Exiting the Sleep mode

The Sleep mode is exit according *Section : Exiting from a low-power mode*.

Refer to *Table 67: Sleep mode* for more details on how to exit the Sleep mode.

Table 67. Sleep mode

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®-M33 system control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex®-M33 System Control register.

Table 67. Sleep mode (continued)

Sleep-now mode	Description
Mode exit	If WFI or return from ISR was used for entry Interrupt: refer to *Table 104: STM32L552xx and STM32L562xx vector table* If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to *Section 17.3: EXTI functional description* If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to *Table 104: STM32L552xx and STM32L562xx vector table* or Wakeup event: refer to *Section 17.3: EXTI functional description*
Wakeup latency	None

8.4.6 Low-power sleep mode (LP sleep)

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 to *Section : Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex®-M33 system control register is clear.

Refer to *Table 68: 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 to *Section : Exiting from a 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 68: Low-power sleep* for details on how to exit the Low-power sleep mode.

Table 68. 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®-M33 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®-M33 System Control register.
Mode exit	If WFI or Return from ISR was used for entry Interrupt: refer to *Table 104: STM32L552xx and STM32L562xx vector table* If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to *Section 17.3: EXTI functional description* If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to *Table 104: STM32L552xx and STM32L562xx vector table* Wakeup event: refer to *Section 17.3: EXTI functional description* After exiting the Low-power sleep mode, the MCU is in Low-power run mode.
Wakeup latency	None

8.4.7 Stop 0 mode

The Stop 0 mode is based on the Cortex®-M33 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 MSI, the HSI16 and the HSE oscillators are disabled. Some peripherals with the wakeup capability (I2Cx (x=1,2,3), 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 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 into a low-power mode*, when the SLEEPDEEP bit in the Cortex®-M33 System Control register is set.

Refer to *Table 69: 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 39.3: IWDG functional description*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the *RCC Backup domain control register (RCC_BDCR)*.
- Internal RC oscillator (LSI): LSI clock or LSI clock divided by 128, this is configured by the LSION and LSIPRE bits in the *RCC control/status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the *RCC Backup domain control register (RCC_BDCR)*.

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, LPTIM2, I2Cx (x=1,2,3,4) U(S)ARTx(x=1,2...5), LPUART.

The DACx (x=1,2), the OPAMPs and the comparators can be used in Stop 0 mode, the PVMx (x=1,2,3,4) 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), 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 into a low-power mode*.

Refer to *Table 69: 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 bit STOPWUCK is set in *RCC clock configuration register (RCC_CFGR)*. The MSI oscillator is selected as system clock if the bit STOPWUCK is cleared. The wakeup time is shorter when HSI16 is selected as wakeup system clock. The MSI selection allows wakeup at higher frequency, up to 48 MHz.

When exiting the Stop 0 mode, the MCU is either in Run mode (Range 0, Range 1 or Range 2) or in Low-power run mode if the bit LPR is set in the PWR_CR1 register.

Table 69. 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®-M33 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®-M33 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 EXTI rising edge pending register (EXTI_RPR2)), 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 104: STM32L552xx and STM32L562xx vector table*. If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in Event mode. Refer to *Section 17.3: EXTI functional description*. 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 104: STM32L552xx and STM32L562xx vector table*. Wakeup event: refer to *Section 17.3: EXTI functional description*
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and Flash wakeup time from Stop 0 mode.

8.4.8 Stop 1 mode

The Stop 1 mode is the same as Stop 0 mode except that the main regulator is OFF, and only the low-power regulator is ON. Stop 1 mode can be entered from Run mode and from Low-power run mode.

Refer to *Table 70: Stop 1 mode* for details on how to enter and exit Stop 1 mode.

Table 70. 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®-M33 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®-M33 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 EXTI rising edge pending register (EXTI_RPR1)), 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 104: STM32L552xx and STM32L562xx vector table*. If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in Event mode. Refer to *Section 17.3: EXTI functional description*. 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 104: STM32L552xx and STM32L562xx vector table*. Wakeup event: refer to *Section 17.3: EXTI functional description*
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and regulator wakeup time from low-power mode + Flash wakeup time from Stop 1 mode.

8.4.9 Stop 2 mode

The Stop 2 mode is based on the Cortex®-M33 Deepsleep mode combined with peripheral clock gating. In Stop 2 mode, all clocks in the V_{CORE} domain are stopped, the PLL, the MSI, the HSI16 and the HSE oscillators are disabled. Some peripherals with wakeup capability (I2C3 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 and register contents are preserved.

The BOR is always available in Stop 2 mode. The consumption is increased when thresholds higher than V_{BOR0} are used.

Note: The comparators outputs, the LPUART outputs and the LPTIM1 outputs are forced to low speed (OSPEEDy=00) during the Stop 2 mode.

I/O states in Stop 2 mode

In the Stop 2 mode, all I/O pins keep the same state as in the Run mode.

Entering Stop 2 mode

The Stop 2 mode is entered according *Section : Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex®-M33 System Control register is set.

Refer to *Table 71: Stop 2 mode* for details on how to enter the Stop 2 mode.

Stop 2 mode can only be entered from Run mode. It is not possible to enter Stop 2 mode from the Low-power run mode.

If Flash memory programming is ongoing, the Stop 2 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop 2 mode entry is delayed until the APB access is finished.

In Stop 2 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 39.3: IWDG functional description* in *Section 39: Independent watchdog (IWDG)*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the *RCC Backup domain control register (RCC_BDCR)*.
- Internal RC oscillator (LSI): LSI clock or LSI clock divided by 128, this is configured by the LSION and LSIPRE bits in the *RCC control/status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the *RCC Backup domain control register (RCC_BDCR)*.

Several peripherals can be used in Stop 2 mode and can add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock: LPTIM1, I2C3, LPUART.

The comparators can be used in Stop 2 mode, the PVMx (x=1,2,3,4) and the PVD as well. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx, OPAMPx, DACx, temperature sensor and VREFBUF buffer can consume power during Stop 2 mode, unless they are disabled before entering this mode.

All the peripherals which cannot be enabled in Stop 2 mode must be either disabled by clearing the enable bit in the peripheral itself, or put under reset state by setting the corresponding bit in the *RCC AHB1 peripheral reset register (RCC_AHB1RSTR)*, *RCC AHB2 peripheral reset register (RCC_AHB2RSTR)*, *RCC AHB3 peripheral reset register (RCC_AHB3RSTR)*, *RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)*, *RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)*, *RCC APB2 peripheral reset register (RCC_APB2RSTR)*.

Exiting Stop 2 mode

The Stop 2 mode is exit according to *Section : Exiting from a low-power mode*.

Refer to *Table 71: Stop 2 mode* for details on how to exit Stop 2 mode.

When exiting Stop 2 mode by issuing an interrupt or a wakeup event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in *RCC clock configuration register (RCC_CFGR)*. The MSI oscillator is selected as system clock if the bit STOPWUCK is cleared. The wakeup time is shorter when HSI16 is selected as wakeup system clock. The MSI selection allows wakeup at higher frequency, up to 48 MHz.

When exiting the Stop 2 mode, the MCU is in Run mode (Range 0, Range 1 or Range 2 depending on VOS bit in PWR_CR1).

Table 71. Stop 2 mode

Stop 2 mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex®-M33 system control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “010” in PWR_CR1
	On return from ISR while: – SLEEPDEEP bit is set in Cortex®-M33 System Control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = “010” in PWR_CR1
	Note: To enter Stop 2 mode, all EXTI Line pending bits (in EXTI rising edge pending register (EXTI_RPR2)), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop 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 104: STM32L552xx and STM32L562xx vector table*. If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in Event mode. Refer to *Section 17.3: EXTI functional description*. 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 104: STM32L552xx and STM32L562xx vector table*. Any EXTI Line configured in Event mode. Refer to *Section 17.3: EXTI functional description*.
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and regulator wakeup time from low-power mode + Flash wakeup time from Stop 2 mode.

8.4.10 Standby mode

The Standby mode permits the achievement of the lowest power consumption with BOR. It is based on the Cortex®-M33 Deepsleep mode, with the voltage regulators disabled (except when SRAM2 content is preserved). The PLL, the HSI16, the MSI 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 21*). SRAM2 content can be can be partially or fully preserved depending on RRS[1:0] bits configuration in PWR_CR3. 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 IO's are by default in floating state. If the APC bit of PWR_CR3 register has been set, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x=A,B,C,D,E,F,G,H)), or with a pull-down (refer to PWR_PDCRx registers (x=A,B,C,D,E,F,G,H)), or can be kept in analog state if none of the PWR_PUCRx or PWR_PDCRx register has been set. The pull-down configuration has highest priority over pull-up configuration in case both PWR_PUCRx and PWR_PDCRx are set for the same IO.

Some I/Os (listed in *Section 11.3.1: General-purpose I/O (GPIO)*) are used for JTAG/SW debug and can only be configured to their respective reset pull-up or pull-down state during Standby mode setting their respective bit in the PWR_PUCRx or PWR_PDCRx registers to '1', or to be configured to floating state if the bit is kept at '0'.

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 tamper inputs are available.

Entering Standby mode

The Standby mode is entered according *Section : Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex®-M33 System Control register is set.

Refer to *Table 72: 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 39.3: IWDG functional description* in *Section 39: 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): LSI clock or LSI clock divided by 128, this is configured by the LSION and LSIPRE bits 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 exited according *Section : Entering into a 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 72: Standby mode* for more details on how to exit Standby mode.

When exiting Standby mode, I/O's that were configured with pull-up or pull-down during Standby through registers PWR_PUCRx or PWR_PDCRx keep this configuration upon exiting Standby mode until the bit APC of PWR_CR3 register has been cleared. Once the bit APC is cleared, they are either configured to their reset values or to the pull-up/pull-down state according the GPIOx_PUPDR registers. The content of the PWR_PUCRx or PWR_PDCRx registers however is not lost and can be re-used for a sub-sequent entering into Standby mode.

Some I/Os (listed in *Section 11.3.1: General-purpose I/O (GPIO)*) are used for JTAG/SW debug and have internal pull-up or pull-down activated after reset so is configured at this reset value as well when exiting Standby mode.

For IO's, with a pull-up or pull-down pre-defined after reset (some JTAG/SW IO's) or with GPIOx_PUPDR programming done after exiting from Standby, in case those programming is different from the PWR_PUCRx or PWR_PDCRx programmed value during Standby, both a pull-down and pull-up are applied until the bit APC is cleared, releasing the PWR_PUCRx or PWR_PDCRx programmed value.

Table 72. Standby mode

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M33 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®-M33 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

8.4.11 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 MSI, the LSI and the HSE oscillators are also switched off.

SRAM1, SRAM2 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, are by default in floating state. If the APC bit of PWR_CR3 register has been set, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x=A,B,C,D,E,F,G,H), or with a pull-down (refer to PWR_PDCRx registers (x=A,B,C,D,E,F,G,H)), or can be kept in analog state if none of the PWR_PUCRx or PWR_PDCRx register has been set. The pull-down configuration has highest priority over pull-up configuration in case both PWR_PUCRx and PWR_PDCRx are set for the same IO. However this configuration is lost when exiting the Shutdown mode due to the power-on reset.

Some I/Os (listed in *Section 11.3.1: General-purpose I/O (GPIO)*) are used for JTAG/SW debug and can only be configured to their respective reset pull-up or pull-down state during Standby mode setting their respective bit in the PWR_PUCRx or PWR_PDCRx registers to ‘1’, or to be configured to floating state if the bit is kept at ‘0’.

The RTC outputs on PC13 are functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. Five wakeup pins (WKUPx, x=1,2...5) and the three RTC tampers are available.

Entering Shutdown mode

The Shutdown mode is entered according *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex®-M33 System Control register is set.

Refer to *Table 73: 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 from a 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 73: Shutdown mode* for more details on how to exit Shutdown mode.

When exiting Shutdown mode, I/Os that were configured with pull-up or pull-down during Shutdown through registers PWR_PUCRx or PWR_PDCRx lose their configuration and are

configured in floating state or to their pull-up pull-down reset value (for some I/Os listed in *Section 11.3.1: General-purpose I/O (GPIO)*).

Table 73. Shutdown mode

Shutdown mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M33 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®-M33 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

8.4.12 Auto-wakeup from a low-power mode

The RTC can be used to wakeup the MCU from a low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop (0, 1 or 2) or Standby mode at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the *RCC Backup domain control register (RCC_BDCR)*:

- Low-power 32.768 kHz external crystal oscillator (LSE OSC)
  This clock source provides a precise time base with very low-power consumption.
- Low-power internal RC Oscillator (LSI)
  This clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC Oscillator is designed to add minimum power consumption.

To wakeup from Stop mode with an RTC event (alarm, wake-up timer, timestamp), it is necessary to:

- Configure the EXTI Line 17 for non-secure interrupt and EXTI Line 18 for secure interrupts to be sensitive to rising edge
- Configure the RTC to generate the RTC event (alarm, wake-up timer, timestamp)

To wakeup from Standby mode, there is no need to configure the EXTI Line 17 or EXTI Line 18.

8.5 PWR TrustZone security

When the TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register, some PWR register fields can be secured against non-secure access.

The PWR TrustZone security allows to secure the following features through the security configuration register PWR_SECCFGR:

- Low-power mode
- Wake-up (WKUP) pins
- Voltage detection and monitoring
- VBAT mode

Other PWR configuration bits are secure when:

- The system clock selection is secure in RCC, the voltage scaling (VOS) configuration is secure
- A GPIO is configured as secure, its corresponding bit for Pull-up/Pull-down configuration in Standby mode is secure
- The RTC is secure, the backup domain write protection DBP bit in PWR_CR1 register is secure.
- The UCPD is secure, the UCPD_DBDIS and UCPD_STDBY bits are secure in the PWR_CR3 register.

Table 74 gives a summary of the PWR secured bits following the security configuration bit in the PWR_SECCFGR register. When one security configuration bit is set, some configuration bits are secured. The PWR registers may contain secure and non-secure bits:

- Secured bits: read and write operations are only allowed by a secure access. Non-secure read or write accesses are RAZ/WI. There is no illegal access event generated.
- Non-secure bits: no restriction. Read and write operations are allowed by both secure and non-secure accesses.

A non-secure write access to PWR_SECCFGR register is WI and generates an illegal access event. An illegal access interrupt is generated if the PWR illegal access interrupt is enabled in the TZIC_IER2 register. There is no restriction for non-secure read access.

When the TrustZone security is disabled (TZEN = 0 in FLASH_OPTR register), all registers are non-secure. The PWR_SECCFGR secure register and security status registers are RAZ/WI.

Table 74. PWR Security configuration summary

Secure configuration register	Security configuration bit	Secured bits	Register name	Non-secure access on secure bits
PWR_SECCFGR	NA(1)	-	PWR_SECCFGR	Read is OK. WI and illegal access event
PWR_SECCFGR	At least one bit is set	PRIV	PWR_PRIVCFGR	Read is OK. WI and illegal access event(2)

Table 74. PWR Security configuration summary

Secure configuration register	Security configuration bit	Secured bits	Register name	Non-secure access on secure bits
PWR_SECCFGR	LPMSEC	LPMS[1:0], LPR	PWR_CR1	RAZ/WI
		RRS[1:0]	PWR_CR3	
		CSBF, CWUFx	PWR_SCR	WI
	WUPxSEC or LPMSEC(1)	WUPx	PWR_CR3	RAZ/WI
		WUPPx	PWR_CR4	
	VDMSEC	All bits in PWR_CR2	PWR_CR2	
		ULPMEN	PWR_CR3	
		SMPSBYP, EXTSMPSEN, SMPSFSTEN, SMPSLPEN	PWR_CR4	RAZ/WI
	VBSEC	DBP	PWR_CR1	
		VBRS, VBE	PWR_CR4	
	APCSEC	APC	PWR_CR3	
TZSC_SECFGR	UCPD1SEC	UCPD_DBDIS and UCPD_STDBY	PWR_CR3	RAZ/WI
RCC_SECCFGR	CKSYSSEC	VOS[1:0]	PWR_CR1	
GPIOx_SECCFGR (x=A,B..H)	SECy (y=0..15)	PUy (y=0..15)	PWR_PUCRx (x = A, B..H)	
		PDy (y=0..15)	PWR_PDCRx (x = A, B..H)	

1. PWR_SECCFGR register is always secure.
2. Illegal access event is generated only when the PWR_PRIVCFGR is secure.

8.5.1 PWR Privileged and Unprivileged modes

The PWR registers can be can be read and written by privileged and unprivileged accesses depending on PRIV bit in PWR_PRIVCFGR register.

- When the PRIV bit is reset, all PWR registers could be read and written by both privileged or unprivileged access.
- When the PRIV bit is set, all PWR registers could be read and written by privileged access only (except PWR_SR1, PWR_SR2 and PWR_SECFGR registers). Unprivileged access to a privileged registers is RAZ/WI.

8.6 PWR registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

8.6.1 Power control register 1 (PWR_CR1)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on PWR_SECCFGR and RCC_SECCFGR configuration registers. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

This register is reset after wakeup from Standby mode.

Address offset: 0x00

Reset value: 0x0000 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 run mode (LPR).

Note: Stop 2 mode cannot be entered when LPR bit is set. Stop 1 is entered instead.

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:9 **VOS[1:0]**: Voltage scaling range selection

00: Range 0
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: Stop 2 mode
011: Standby mode
1xx: Shutdown mode

Note: If LPR bit is set, Stop 2 mode cannot be selected and Stop 1 mode shall be entered instead of Stop 2.

Note: In Standby mode, SRAM2 can be preserved or not, depending on RRS bit configuration in PWR_CR3.

8.6.2 Power control register 2 (PWR_CR2)

When the system is secure (TZEN =1), this register is protected against non-secure access when VDMSEC=1 in the PWR_SECCFGR register. A non-secure read/write access is RAZ/WI and generates and illegal access event.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

Address offset: 0x04

Reset value: 0x0000 0000

This register is reset when exiting the 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.	USV	IOSV	Res.	PVME4	PVME3	PVME2	PVME1	PLS[2:0]			PVDE
					rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **USV**: V_{DDUSB} USB supply valid
This bit is used to validate the V_{DDUSB} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the USB peripheral. If V_{DDUSB} is not always present in the application, the PVM can be used to determine whether this supply is ready or not.
0: V_{DDUSB} is not present. Logical and electrical isolation is applied to ignore this supply.
1: V_{DDUSB} is valid.

Bit 9 **IOSV**: V_{DDIO2} Independent I/Os supply valid
This bit is used to validate the V_{DDIO2} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PG[15:2]. If V_{DDIO2} is not always present in the application, the PVM can be used to determine whether this supply is ready or not.
0: V_{DDIO2} is not present. Logical and electrical isolation is applied to ignore this supply.
1: V_{DDIO2} is valid.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **PVME4**: Peripheral voltage monitoring 4 enable: V_{DDA} vs. 1.8 V
0: PVM4 (V_{DDA} monitoring vs. 1.8 V threshold) disable.
1: PVM4 (V_{DDA} monitoring vs. 1.8 .V threshold) enable.

Bit 6 **PVME3**: Peripheral voltage monitoring 3 enable: V_{DDA} vs. 1.62V
0: PVM3 (V_{DDA} monitoring vs. 1.62V threshold) disable.
1: PVM3 (V_{DDA} monitoring vs. 1.62V threshold) enable.

Bit 5 **PVME2**: Peripheral voltage monitoring 2 enable: V_{DDIO2} vs. 0.9V
0: PVM2 (V_{DDIO2} monitoring vs. 0.9V threshold) disable.
1: PVM2 (V_{DDIO2} monitoring vs. 0.9V threshold) enable.

Bit 4 **PVME1**: Peripheral voltage monitoring 1 enable: V_{DDUSB} vs. 1.2V
0: PVM1 (V_{DDUSB} monitoring vs. 1.2V threshold) disable.
1: PVM1 (V_{DDUSB} monitoring vs. 1.2V threshold) enable.

Bits 3:1 **PLS[2:0]**: Power voltage detector level selection.
These bits select the voltage threshold detected by the power voltage detector:
000: V_{PVD0} around 2.0 V
001: V_{PVD1} around 2.2 V
010: V_{PVD2} around 2.4 V
011: V_{PVD3} around 2.5 V
100: V_{PVD4} around 2.6 V
101: V_{PVD5} around 2.8 V
110: V_{PVD6} around 2.9 V
111: External input analog voltage PVD_IN (compared internally to VREFINT)

Note: These bits are write-protected when the bit PVDL (PVD Lock) is set in the SYSCFG_CBR register.
These bits are reset only by a system reset.

Bit 0 **PVDE**: Power voltage detector enable
0: Power voltage detector disable.
1: Power voltage detector enable.

Note: This bit is write-protected when the bit PVDL (PVD Lock) is set in the SYSCFG_CBR register.
This bit is reset only by a system reset.

8.6.3 Power control register 3 (PWR_CR3)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on PWR_SECCFGR and TZSC_SECCFGR configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

Access: Additional APB cycles are needed to access this register versus a standard APB access (3 for a write and 2 for a read).

This register is not reset when exiting Standby modes and with the PWRRST bit in the RCC_APB1RSTR1 register.

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.	UCPD_DBDIS	UCPD_STDBY	Res.	ULPMEN	APC	RRS[1:0]		Res.	Res.	Res.	EWUP5	EWUP4	EWUP3	EWUP2	EWUP1
	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bit 14 **UCPD_DBDIS**: USB Type-C and power delivery dead dattery disable

After exiting reset, the USB Type-C “dead battery” behavior is enabled, which may have a pulldown 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 UCPD (which should therefore be initialized before doing the disable).

0: enable USB Type-C dead battery pull-down behavior on UCPDx_CC1 and UCPDx_CC2 pins

1: disable USB Type-C dead battery pull-down behavior on UCPDx_CC1 and UCPDx_CC2 pins

Bit 13 **UCPD_STDBY**: USB Type-C and power delivery Standby mode

When set, this bit allows to memorize the UCPD configuration in Standby mode.

This bit must be written to ‘1’ just before entering Standby mode when using UCPD, and it must be written to 0 after exiting the Standby mode and before writing any UCPD registers.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **ULPMEN**: Ultra-low-power mode enable

When this bit is set, the BOR and PVD are in Ultra-low-power mode during Stop 2 and Standby mode in order to further reduce the current consumption.

0: BORL, BORH and PVD operating in Default mode.

1: Enable Ultra-low-power mode for BORL, BORH and PVD during Stop 2 and Standby mode.

In Stop 2 mode, ULPMEN can be set if BORH and PVD are enabled, otherwise there is no power consumption optimization.

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, instead the I/Os are in Floating mode during Standby or configured according GPIO controller GPIOx_PUPDR register during Run mode.

Bits 9:8 **RRS[1:0]**: SRAM2 retention in Standby mode

00: SRAM2 is powered off in Standby mode (SRAM2 content is lost).

01: Full SRAM2 is powered by the low-power regulator in Standby mode (Full SRAM2 content is kept).

10: Only the upper 4 Kbytes of SRAM2 are powered by the low-power regulator in Standby mode (upper 4 Kbytes of SRAM2 content 0x2003 F000 - 0x2003 FFFF is kept).

11: Reserved

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 WUPP5 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 WUPP4 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 WUPP3 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 WUPP2 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 WUPP1 bit in the PWR_CR4 register.

8.6.4 Power control register 4 (PWR_CR4)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on PWR_SECCFGR configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with the PWRRST bit in the RCC_APB1RSTR1 register.

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
SMPSLPEN	SMPSFSTEN	EXTSMPSEN	SMPSBYP	Res.	Res.	VBRS	VBE	Res.	Res.	Res.	WUPP5	WUPP4	WUPP3	WUPP2	WUPP1
rw	rw	rw	rw			rw	rw				rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SMPSLPEN**: Enable SMPS low-power mode

When this enable the SMPS low-power mode. When set, it is forbidden to modify the voltage scaling configuration.

0: SMPS low-power mode disable

1: SMPS low-power mod enable

Bit 14 **SMPSFSTEN**: Enable SMPS fast soft start

0: SMPS fast soft start disable

1: SMPS fast soft start enable.

Bit 13 **EXTSMPSEN**: Enable external SMPS mode
This bit is used for external SMPS mode, it must be set when the external SMPS switch is closed.
0: Disable the external SMPS mode
1: Enable external SMPS mode

Bit 12 **SMPSBYP**: SMPS Bypass mode
This bit is used to force the SMPS step down converter in Bypass mode.
0: SMPS Bypass mode disable
1: SMPS Bypass mode enable

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **VBRS**: VBAT battery charging resistor selection
0: Charge VBAT through a 5 kOhms resistor
1: Charge VBAT through a 1.5 kOhms resistor

Bit 8 **VBE**: VBAT battery charging enable
0: VBAT battery charging disable
1: VBAT battery charging enable

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **WUPP5**: 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 **WUPP4**: 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 **WUPP3**: 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 **WUPP2**: 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 **WUPP1**: 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)

8.6.5 Power status register 1 (PWR_SR1)

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

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.

Address offset: 0x10

Reset value: 0x00A0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPSHPRDY	Res.	EXTSMPSRDY	SMPSBYPRDY	Res.	Res.	Res.	SBF	Res.	Res.	Res.	WUF5	WUF4	WUF3	WUF2	WUF1
r		r	r				r				r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SMPSHPRDY**: SMPS High-power mode ready
This bit is set when the SMPS step down converter is in High-power mode.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **EXTSMPSRDY**: External SMPS mode ready
This bit is set when the main regulator is ready and can be switched in external SMPS mode. When set, the external SMPS switch can be closed.

Bit 12 **SMPSBYPRDY**: SMPS BYPASS ready
This bit is set when the SMPS step down converter is in Bypass mode. It is cleared when the SMPS exits the Bypass mode when it is switched to High-power or Low-power mode.

Bits 11:9 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.

8.6.6 Power status register 2 (PWR_SR2)

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

This register is partially reset when exiting Standby/Shutdown modes.

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
PVMO4	PVMO3	PVMO2	PVMO1	PVDO	VOSF	REGLPF	REGLPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PVMO4**: Peripheral voltage monitoring output: VDDA vs. 1.8 V
0: VDDA voltage is above PVM4 threshold (around 1.8 V).
1: VDDA voltage is below PVM4 threshold (around 1.8 V).
Note: PVMO4 is cleared when PVM4 is disabled (PVME4 = 0). After enabling PVM4, the PVM4 output is valid after the PVM4 wakeup time.

Bit 14 **PVMO3**: Peripheral voltage monitoring output: VDDA vs. 1.62 V
0: VDDA voltage is above PVM3 threshold (around 1.62 V).
1: VDDA voltage is below PVM3 threshold (around 1.62 V).
Note: PVMO3 is cleared when PVM3 is disabled (PVME3 = 0). After enabling PVM3, the PVM3 output is valid after the PVM3 wakeup time.

Bit 13 **PVMO2**: Peripheral voltage monitoring output: VDDIO2 vs. 0.9 V
0: VDDIO2 voltage is above PVM2 threshold (around 0.9 V).
1: VDDIO2 voltage is below PVM2 threshold (around 0.9 V).
Note: PVMO2 is cleared when PVM2 is disabled (PVME2 = 0). After enabling PVM2, the PVM2 output is valid after the PVM2 wakeup time.

Bit 12 PVMO1: Peripheral voltage monitoring output: VDDUSB vs. 1.2 V
0: VDDUSB voltage is above PVM1 threshold (around 1.2 V).
1: VDDUSB voltage is below PVM1 threshold (around 1.2 V).
Note: PVMO1 is cleared when PVM1 is disabled (PVME1 = 0). After enabling PVM1, the PVM1 output is valid after the PVM1 wakeup time.

Bit 11 **PVDO**: Power voltage detector output
0: VDD is above the selected PVD threshold
1: VDD 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 from 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

Bits 7:0 Reserved, must be kept at reset value.

8.6.7 Power status clear register (PWR_SCR)

When the system is secure (TZEN =1), this register is protected against non-secure access when LPMSEC=1 in the PWR_SECCFGR register. A non-secure read/write access is RAZ/WI and generates and illegal access event.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

Access: 3 additional APB cycles are needed to write this register vs. a standard APB write.

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.	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.

8.6.8 Power Port A pull-up control register (PWR_PUCRA)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Address offset: 0x20.

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port A pull-up bit y (y=0..15)
When set, this bit activates the pull-up on PA[y] when APC bit is set in PWR_CR3 register. If the corresponding PDy bit is also set, the pull-up is not activated and the pull-down is activated instead with highest priority.

8.6.9 Power Port A pull-down control register (PWR_PDCRA)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

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
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 31:14 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port A pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PA[y] when APC bit is set in PWR_CR3 register.

8.6.10 Power Port B pull-up control register (PWR_PUCRB)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

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).

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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port 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.

8.6.11 Power Port B pull-down control register (PWR_PDCRB)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

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).

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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port B pull-down bit y (y=0..15)
When set, this bit activates the pull-down on PB[y] when APC bit is set in PWR_CR3 register.

8.6.12 Power Port C pull-up control register (PWR_PUCRC)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

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).

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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port 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.

8.6.13 Power Port C pull-down control register (PWR_PDCRC)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the

PWR_PRIVCFGR register.

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).

Address offset: 0x34.

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port 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.

8.6.14 Power Port D pull-up control register (PWR_PUCRD)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

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.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port 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.

8.6.15 Power Port D pull-down control register (PWR_PDCRD)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

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).

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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port 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.

8.6.16 Power Port E pull-up control register (PWR_PUCRE)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Address offset: 0x20.

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port 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.

8.6.17 Power Port E pull-down control register (PWR_PDCRE)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port 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.

8.6.18 Power Port F pull-up control register (PWR_PUCRF)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Address offset: 0x48.

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port 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 PFy bit is also set.

8.6.19 Power Port F pull-down control register (PWR_PDCRF)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Address offset: 0x4C.

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port 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.

8.6.20 Power Port G pull-up control register (PWR_PUCRG)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Address offset: 0x50.

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port G pull-up bit y (y=0..15)
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 PGy bit is also set.

8.6.21 Power Port G pull-down control register (PWR_PDCRG)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Address offset: 0x54.

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port G pull-down bit y (y=0..15)
When set, this bit activates the pull-down on PG[y] when APC bit is set in PWR_CR3 register.

8.6.22 Power Port H pull-up control register (PWR_PUCRH)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Address offset: 0x58.

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PU[15:0]**: Port H pull-up bit y (y=0..15)

When set, this bit activates the pull-up on PH[y] when APC bit is set in PWR_CR3 register. The pull-up is not activated if the corresponding PHy bit is also set.

8.6.23 Power Port H pull-down control register (PWR_PDCRH)

When the system is secure (TZEN =1), some register fields are protected against non-secure access depending on GPIO secure bit configuration. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register is protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR 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).

This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PD[15:0]**: Port H pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PH[y] when APC bit is set in PWR_CR3 register.

8.6.24 Power secure configuration register (PWR_SECCFGR)

When the system is secure (TZEN =1), this register provides write access security and can be written only when the access is secure. A non-secure write access is WI and generates an illegal access event. There are no read restrictions

When the system is not secure (TZEN=0), this register is RAZ/WI.

This register can be protected against non-privileged access when PRIV=1 in the PWR_PRIVCFGR register.

Address offset: 0x78.

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	APCSEC	VBSEC	VDMSEC	LPMSEC	Res.	Res.	Res.	WUP5SEC	WUP4SEC	WUP3SEC	WUP2SEC	WUP1SEC
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **APCSEC**: APC security

0: APC bit in PWR_CR3 can be read and written by secure or non-secure access.

1: APC bit in PWR_CR3 can be read and written by secure access only

Bit 10 **VBSEC**: Voltage battery security

0: VBRS and VBE bits in PWR_CR4 and DPB bit in PWR_CR1 can be read and written by secure or non-secure access.

1: VBRS and VBE bits in PWR_CR4 and DPB bit in PWR_CR1 can be read and written by secure access only.

Refer to *Table 74* for full list of secured bits.

Bit 9 **VDMSEC**: Voltage detection and monitoring security

0: PWR_CR2 and some bit fields in PWR_CR3 and PWR_CR4 registers can be read and written by secure or non-secure access.

1: PWR_CR2 and some bit fields in PWR_CR3 and PWR_CR4 registers can be read and written by secure access only.

Refer to *Table 74* for full list of secured bits.

Bit 8 **LPMSEC**: Low-power mode security

0: Low-power mode and WKUPx related bit field in PWR_CR1, PWR_CR3, PWR_CR4 and PWR_SCR can be read and written by secure or non-secure access.

1: Low-power mode and WKUPx related bit field in PWR_CR1, PWR_CR3, PWR_CR4 and PWR_SCR can be read and written by secure access only

Note: This bit has a priority over WKUPxSEC bit.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **WUP5SEC**: WKUP5 pin security

0: The bits related to the WKUP5 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure or non-secure access.

1: The bits related to the WKUP5 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure access only

Bit 3 **WUP4SEC**: WKUP4 pin security

0: The bits related to the WKUP4 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure or non-secure access.

1: The bits related to the WKUP4 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure access only

Bit 2 **WUP3SEC**: WKUP3 pin security

0: The bits related to the WKUP3 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure or non-secure access.

1: The bits related to the WKUP3 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure access only

Bit 1 **WUP2SEC**: WKUP2 pin security

0: The bits related to the WKUP2 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure or non-secure access.

1: The bits related to the WKUP2 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure access only

Bit 0 **WUP1SEC**: WKUP1 pin security

0: The bits related to the WKUP1 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure or non-secure access.

1: The bits related to the WKUP1 wakeup pin in PWR_CR3 and PWR_CR4 can be read and written by secure access only

8.6.25 Power privilege configuration register (PWR_PRIVCFGR)

This register can be read by both privileged and unprivileged access.
When the system is secure (TZEN =1),this register can be read by secure and non-secure access. It is write-protected against non-secure write access when at least one bit is set in the PWR_SECCFGR register

Address offset: 0x80.

Reset value: 0x0000 0000

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PRIV**: Privilege protection

Set and reset by software.

This bit can be read by both privileged or unprivileged access. when set, it can only be cleared by a privileged access.

0: All PWR registers can be read and written with privileged or non-privileged access.

1: All PWR registers (except PWR_SR1, PWR_SR2 and PWR_SECFGR registers) can be read and written only with privileged access. An unprivileged access to PWR registers is RAZ/WI.

If the PWR is not secure, the PRIV bit can be written by a secure or non-secure privileged access.

If TrustZone security is enabled (TZEN = 1), if the PWR is secure, the PRIV bit can be written only by a secure privileged access:

– A non-secure write access generates an illegal access event and write is ignored.
– A secure unprivileged write access on PRIV bit is ignored.

8.6.26 PWR register map and reset values

Table 75. PWR register map and reset values

Offset	Register name reset value	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.	Res.	LPMS [2:0]		
	Reset value																		0				0	1	0						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.	USV	IOSV	Res.	PVME4	PVME3	PVME2	PVME1	PLS [2:0]		PVDE	
	Reset value																						0	0		0	0	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.	Res.	UCPD_DBDIS	UCPD_STDBY	Res.	ULPMEN	APC	RRS[1:0]		Res.	Res.	Res.	EWUP5	EWUP4	EWUP3	EWUP2	EWUP1
	Reset value																		0	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.	SMPSLPEN	SMPSFSTEN	EXTSMPSEN	SMPSBYP		Res.	VBRS	VBE	Res.	Res.	Res.	WUPP5	WUPP4	WUPP3	WUPP2	WUPP1
	Reset value																	0	0	0	0			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.	SMPSHPRDY	Res.	EXTSMPSRDY	SMPSBYPRDY	Res.	Res.	Res.	SBF	Res.	Res.	Res.	WUF5	WUF4	WUF3	WUF2	WUF1
	Reset value																	0		0	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.	PVMO4	PVMO3	PVMO2	PVMO1	PVDO	VOSF	REGLPF	REGLPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	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	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
0x24	PWR_PDCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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

Table 75. PWR register map and reset values (continued)

Offset	Register name reset value	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	PWR_PDCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
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	PD1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	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.	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
0x54	PWR_PDCRG	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
0x58	PWR_PUCRH	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
0x5C	PWR_PDCRH	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
0x78	PWR_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APCSEC	VBSEC	VDMSEC	LPMSEC	Res.	Res.	Res.	WUP5SEC	WUP4SEC	WUP3SEC	WUP2SEC	WUP1SEC
	Reset value																					0	0	0	0				0	0	0	0	0
0x80	PWR_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV
	Reset value																																0

Refer to *Section 2.3 on page 87* for the register boundary addresses.

9 Reset and clock control (RCC)

9.1 Reset

There are three types of reset:

- a system reset
- a power reset
- a Backup domain reset

9.1.1 Power reset

A power reset is generated when one of the following events occurs:

- a Brownout reset (BOR)
- when exiting from Standby mode
- when exiting from Shutdown mode

A Brownout reset, including power-on or power-down reset (POR/PDR), sets all registers to their reset values except the ones in the Backup domain.

When exiting Standby mode, all registers in the V_{CORE} domain are set to their reset value. Registers outside the V_{CORE} domain (RTC, WKUP, IWDG, and Standby/Shutdown modes control) are not impacted.

When exiting Shutdown mode, a Brownout reset is generated, resetting all registers except those in the Backup domain.

9.1.2 System reset

A system reset sets all registers to their reset values except the reset flags in *RCC control/status register (RCC_CSR)* and the registers in the Backup domain.

A system reset is generated when one of the following events occurs:

- a low level on the NRST pin (external reset)
- a window watchdog event (WWDG reset)
- an independent watchdog event (IWDG reset)
- a software reset (SW reset) (see *Software reset*)
- a low-power mode security reset (see *Low-power mode security reset*)
- an option byte loader reset (see *Option byte loader reset*)
- a Brownout reset

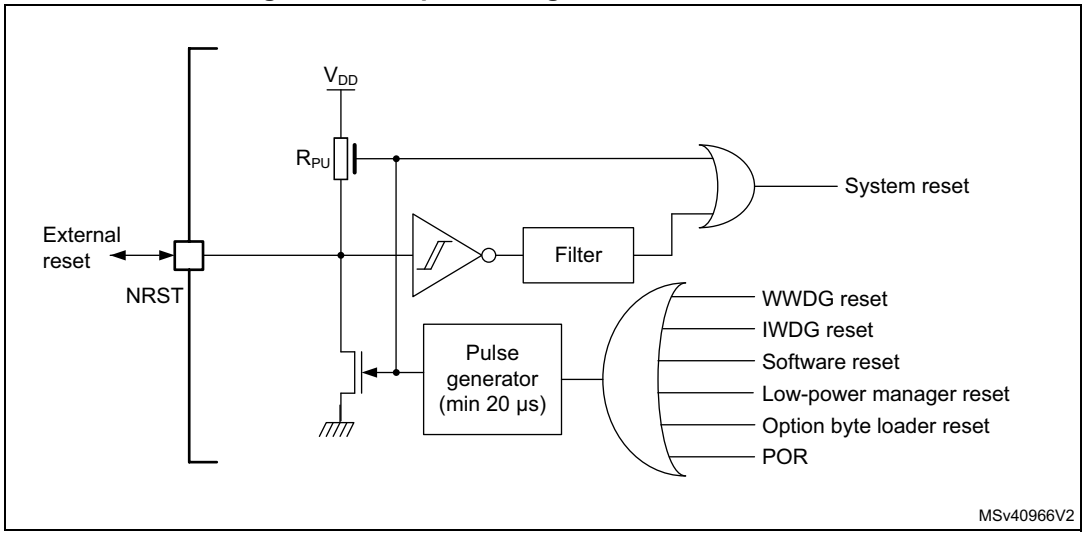
The reset source can be identified by checking the reset flags in *RCC control/status register (RCC_CSR)*.

These sources act on the NRST pin and this pin is always kept low during the delay phase. The reset service routine vector is selected via the Boot option bytes.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

In case on an internal reset, the internal pull-up R_{PU} is deactivated in order to save the power consumption through the pull-up resistor.

Figure 29. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in Cortex®-M33 application interrupt and reset control register must be set to force a software reset on the device.

Low-power mode security reset

To avoid 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 any of the following conditions:

- Entering Standby mode: this type of reset is enabled by resetting nRST_STDBY bit in user option bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
- Entering Stop mode: this type of reset is enabled by resetting nRST_STOP bit in user option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.
- Entering Shutdown mode: this type of reset is enabled by resetting nRST_SHDW bit in user option bytes. In this case, whenever a Shutdown mode entry sequence is successfully executed, the device is reset instead of entering Shutdown mode.

For further information on the user option bytes, refer to *Section 6.4.1: Option bytes description*.

Option byte loader reset

The option byte loader reset is generated when the OBL_LAUNCH bit (#27) is set in the FLASH_CR register. This bit is used to launch the option byte loading by software.

9.1.3 Backup domain reset

The Backup domain has two specific resets.

A Backup domain reset is generated when one of the following events occurs:

- a software reset, triggered by setting the BDRST bit in the *RCC Backup domain control register (RCC_BDCR)*
- a V_{DD} or V_{BAT} power on, if both supplies have previously been powered off

A Backup domain reset only affects the LSE oscillator, the RTC, the backup registers, the SRAM2 and the RCC_BDCR register.

9.2 RCC pins and internal signals

Table 76 lists the RCC inputs and output signals connected to package pins or balls.

Table 76. RCC input/output signals connected to package pins or balls

Signal name	Signal type	Description
NRST	I/O	System reset, can be used to provide reset to external devices
OSC32_IN	I	32 kHz oscillator input
OSC32_OUT	O	32 kHz oscillator output
OSC_IN	I	System oscillator input
OSC_OUT	O	System oscillator output
MCO1	O	Output clock 1 for external devices
SAI1_EXTCLK	I	External kernel clock input for SAI1 digital audio interface
SAI2_EXTCLK	I	External kernel clock input for SAI2 digital audio interface

9.3 Clocks

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI16 (high-speed internal)16 MHz RC oscillator clock
- MSI (multispeed internal) RC oscillator clock
- HSE oscillator clock, from 4 to 48 MHz
- PLL clock

The MSI is used as system clock source after startup from reset, configured at 4 MHz.

The devices have the following additional clock sources:

- 32 kHz low-speed internal RC (LSI RC) that 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) that optionally drives the real-time clock (RTCCLK)
- RC 48 MHz internal clock sources (HSI48) to potentially drive the USB FS, the SDMMC 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, APB1 and APB2 domains is 110 MHz.

All the peripheral clocks are derived from their bus clock (HCLK, PCLK1 or PCLK2) except the following ones:

- The 48 MHz clock used for USB FS, SDMMC and RNG, is derived from one of the following sources (selected by software):
  - main PLL VCO (PLL48M1CLK)
  - PLLSAI1 VCO (PLL48M2CLK)
  - MSI clock
  - HSI48 internal oscillator

  When the MSI clock is auto-trimmed with the LSE, it can be used by the USB FS device.

  When available, the HSI48 clock can be coupled to the clock recovery system (CRS) allowing adequate clock connection for the USB FS (crystal less solution).
- The ADCs clock is derived from one of the following sources (selected by software):
  - system clock (SYSCLK)
  - PLLSAI1 VCO (PLLADC1CLK)
- The U(S)ARTs clocks are derived from one of the following sources (selected by software):
  - system clock (SYSCLK)
  - HSI16 clock
  - LSE clock
  - APB1 or APB2 clock (PCLK1 or PCLK2, depending on which APB is mapped to the U(S)ART)

  The wakeup from Stop mode is supported only when the clock is HSI16 or LSE.
- The I2Cs clocks are derived from one of the following sources (selected by software):
  - system clock (SYSCLK)
  - HSI16 clock
  - APB1 clock (PCLK1)

  The wakeup from Stop mode is supported only when the clock is HSI16.
- The SAI1 and SAI2 clocks are derived from one of the following sources (selected by software):
  - an external clock mapped on SAI1_EXTCLK for SAI1 and SAI2_EXTCLK for SAI2
  - PLLSAI1 VCO (PLLSAI1CLK)
  - PLLSAI2 VCO (PLLSAI2CLK)
  - main PLL VCO (PLLSAI3CLK)
  - HSI16 clock
- The DFSDM audio clock which is derived from one of the following sources (selected by software):
  - SAI1 clock
  - HSI clock

  - MSI clock
- The OCTOSPI kernel clock is derived from one of the following sources (selected by software):
  - system clock
  - PLL48M1CLK
  - MSI clock
- The FDCAN kernel clock is derived from one of the following sources (selected by software):
  - PLL48M1CLK
  - PLLSAI1CLK
  - HSE clock
- The low-power timers (LPTIMx) clocks are derived from one of the following sources (selected by software):
  - LSI clock
  - LSE clock
  - HSI16 clock
  - APB1 clock (PCLK1)
  - external clock mapped on LPTIMx_IN1

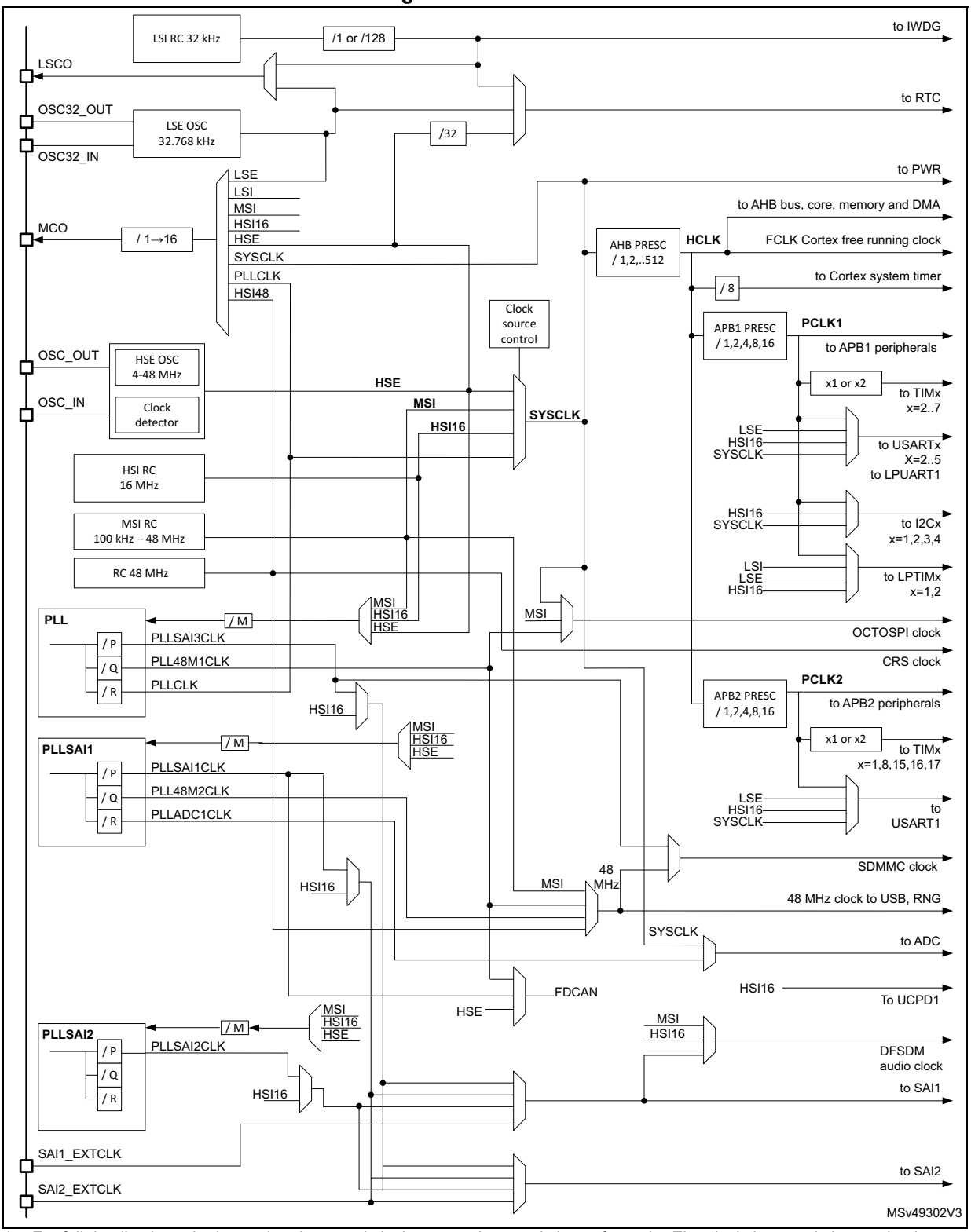
  The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE, or in external clock mode.
- The RTC clock is derived from one of the following sources (selected by software):
  - LSE clock
  - LSI clock
  - HSE clock divided by 32

  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 32 kHz clock.
- The UCPD kernel clock is derived from HSI16 clock. The HSI16 RC oscillator must be enabled prior enabling the UCPD.

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®-M33 free-running clock.

Figure 30. Clock tree



1. For full details about the internal and external clock source characteristics, refer to the Electrical characteristics section in the 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’.

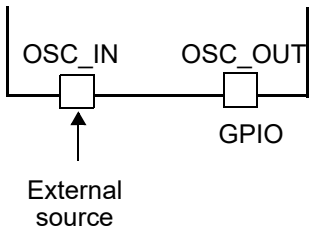
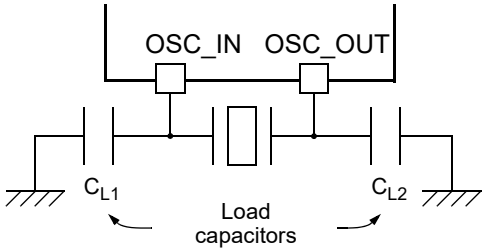
9.3.1 HSE clock

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 31. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	

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 31*. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag in the *RCC 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 *RCC clock interrupt enable register (RCC_CIER)*.

The HSE crystal can be switched on and off using the HSEON bit in the *RCC 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. This mode is selected by setting the HSEBYP and HSEON bits in the *RCC 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) must drive the OSC_IN pin while the OSC_OUT pin can be used a GPIO (see *Figure 31*).

9.3.2 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 or Stop 2). Refer to *Section 9.4: Low-power modes*. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 9.3.11: Clock security system (CSS)*.

Calibration

The RC oscillator frequencies may 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 *RCC internal clock sources calibration register (RCC_ICSCR)*.

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. The HSI16 frequency can be trimmed in the application using the HSITRIM[6:0] in the *RCC internal clock sources calibration register (RCC_ICSCR)*.

For more details on how to measure the HSI16 frequency variation, refer to *Section 9.3.18: Internal/external clock measurement with TIM15/TIM16/TIM17*.

The HSIRDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSI16 RC is stable or not. At startup, the HSI16 RC output clock is not released until this bit is set by hardware.

The HSI16 RC can be switched on and off using the HSION bit in the *RCC clock control register (RCC_CR)*.

The HSI16 signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 9.3.11: Clock security system (CSS) on page 339*.

9.3.3 MSI clock

The MSI clock signal is generated from an internal RC oscillator. Its frequency range can be adjusted by software by using the MSIRANGE[3:0] bits in the *RCC clock control register (RCC_CR)*. Twelve frequency ranges are available: 100 kHz, 200 kHz, 400 kHz, 800 kHz, 1 MHz, 2 MHz, 4 MHz (default value), 8 MHz, 16 MHz, 24 MHz, 32 MHz and 48 MHz.

The MSI clock is used as system clock after restart from Reset, wakeup from Standby and Shutdown low-power modes. After restart from Reset, the MSI frequency is set to its default value 4 MHz. Refer to *Section 9.4: Low-power modes*.

The MSI clock can be selected as system clock after a wakeup from Stop mode (Stop 0, Stop 1 or Stop 2). Refer to *Section 9.4: Low-power modes*. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 9.3.11: Clock security system (CSS)*.

The MSI RC oscillator has the advantage of providing a low-cost (no external components) low-power clock source. In addition, when used in PLL-mode with the LSE, it provides a very accurate clock source that can be used by the USB FS device, and feed the main PLL to run the system at the maximum speed 110 MHz.

The MSIRDY flag in the *RCC clock control register (RCC_CR)* indicates whether the MSI RC is stable or not. At startup, the MSI RC output clock is not released until this bit is set by hardware. The MSI RC can be switched on and off by using the MSION bit in the *RCC clock control register (RCC_CR)*.

Hardware auto calibration with LSE (PLL-mode)

When a 32.768 kHz external oscillator is present in the application, it is possible to configure the MSI in a PLL-mode by setting the MSIPLLEN bit in the *RCC clock control register (RCC_CR)*. When configured in PLL-mode, the MSI automatically calibrates itself thanks to the LSE. This mode is available for all MSI frequency ranges. At 48 MHz, the MSI in PLL-mode can be used for the USB FS device, saving the need of an external high-speed crystal.

Software calibration

The MSI RC oscillator frequency may vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1 % accuracy at an ambient temperature, T_A = 25 °C. After reset, the factory calibration value is loaded in the MSICAL[7:0] bits in the *RCC internal clock sources calibration register (RCC_ICSCR)*. If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. You can trim the MSI frequency in the application by using the MSITRIM[7:0] bits in the RCC_ICSCR register. For more details on how to measure the MSI frequency variation please refer to *Section 9.3.18: Internal/external clock measurement with TIM15/TIM16/TIM17*.

9.3.4 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) as well as SDMMC.

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 USB SOF signal, 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.

For more details on how to configure and use the CRS peripheral please refer to *Section 10: Clock recovery system (CRS)*.

The HSI48RDY flag in the RCC_CRRCR register 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 RCC_CRRCR register.

9.3.5 PLL

The device embeds three PLLs: PLL, PLLSAI1 and PLLSAI2. Each PLL provides up to three independent outputs. The internal PLLs can be used to multiply the HSI16, HSE or MSI output clock frequency. The PLLs input frequency must be between 4 and 16 MHz. The selected clock source for each PLL is divided by a dedicated programmable factor PLLM, PLLSAI1M, PLLSAI2M, from 1 to 8 to provide a clock frequency in the requested input range. Refer to *Figure 30: Clock tree*, *RCC PLL configuration register (RCC_PLLCFGR)*, *RCC PLLSAI1 configuration register (RCC_PLLSAI1CFGR)* and *RCC PLLSAI2 configuration register (RCC_PLLSAI2CFGR)*.

The PLLs 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 clearing PLLON to 0 in *RCC clock control register (RCC_CR)*.
2. Wait until PLLRDY bit is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON bit to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, PLLREN bits in *RCC PLL configuration register (RCC_PLLCFGR)*.

An interrupt can be generated when the PLL is ready, if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

The same procedure is applied for changing the configuration of PLLSAI1 or PLLSAI2:

1. Disable the PLLSAI1/PLLSAI2 by clearing PLLSAI1ON/PLLSAI2ON to 0 in *RCC clock control register (RCC_CR)*.
2. Wait until PLLSAI1RDY/PLLSAI2RDY bit is cleared. The PLLSAI1/PLLSAI2 is now fully stopped.
3. Change the desired parameter.
4. Enable the PLLSAI1/PLLSAI2 again by setting PLLSAI1ON/PLLSAI2ON bit to 1.
5. Enable the desired PLL outputs by configuring PLLSAI1PEN/PLLSAI2PEN, PLLSAI1QEN, PLLSAI1REN bits in *RCC PLLSAI1 configuration register (RCC_PLLSAI1CFGR)* or *RCC PLLSAI2 configuration register (RCC_PLLSAI2CFGR)*.

The PLL output frequency must not exceed 110 MHz.

The enable bit of each PLL output clock (PLLPEN, PLLQEN, PLLREN, PLLSAI1PEN, PLLSAI1QEN, PLLSAI1REN and PLLSAI2PEN) can be modified at any time without stopping the corresponding PLL.

The PLLREN bit cannot be cleared if PLLCLK is used as system clock.

9.3.6 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 *RCC Backup domain control register (RCC_BDCR)*. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the *RCC Backup domain control register (RCC_BDCR)* to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other side. The LSE drive can be decreased to the lower drive capability (LSEDRV = 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 *RCC Backup domain control register (RCC_BDCR)* indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. This mode is selected by setting the LSEBYP and LSEON bits in the *RCC 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 31*.

9.3.7 LSE system clock

The LSE system clock (LSESYS) is generated by RCC to:

- a peripheral when source clock is the LSE as LPTIM, USART, LPUART, TIMER, CRS
- the system in case of one of the LSCOSEL, MCO, MSI PLL mode or CSS on LSE functionality is enabled

By default the LSESYS clock is disabled. To enable the LSESYS clock, proceed as follows:

1. Wait the LSE clock is ready and set the LSEON and LSERDY bits in *RCC Backup domain control register (RCC_BDCR)*.
2. Set the LSESYSEN bit in RCC_BDCR.
3. Wait the LSESYS clock is ready and set the LSESYSRDY bit in RCC_BDCR.

9.3.8 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 either 32 kHz or 250 Hz depending on the LSIPRE bit in *RCC control/status register (RCC_CSR)*. When using the IWDG, only the 32 kHz LSI clock is selected and forced on. For more details, refer to the electrical characteristics section of the datasheet.

The LSI RC can be switched on and off using the LSION bit in the *RCC control/status register (RCC_CSR)*.

The LSI prescaler clock (LSIPRE bit in RCC_CSR) is only taken into account when the LSION bit is reset.

The LSIRDY flag in the *RCC control/status register (RCC_CSR)* indicates if the LSI oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

9.3.9 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

- MSI oscillator
- HSI16 oscillator
- HSE oscillator
- PLL

The system clock maximum frequency is 110 MHz. After a system reset, the MSI oscillator, at 4 MHz, is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *RCC internal clock sources calibration register (RCC_ICSCR)* indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

Clock source switching conditions:

- Switching from HSE or HSI or MSI to PLL with AHB frequency (HCLK) higher than 80 MHz
- Switching from PLL with HCLK higher than 80 MHz to HSE or HSI or MSI

Transition state:

- Set the AHB prescaler HPRE[3:0] bits in RCC_CFGR 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.

9.3.10 Clock source frequency versus voltage scaling

Table 77 gives the different clock source frequencies depending on the product voltage range.

Table 77. Clock source frequency

Product voltage range	Clock frequency			
	MSI	HSI16	HSE	PLL/PLLSAI1/PLLSAI2
Range 0	48 MHz	16 MHz	48 MHz	110 MHz
Range 1	48 MHz	16 MHz	48 MHz	80 MHz
Range 2	24 MHz range	16 MHz	26 MHz	26 MHz

9.3.11 Clock security system (CSS)

CSS can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startups 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 and TIM15/16/17) 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®-M33 NMI (non-maskable interrupt) exception vector.

Note: Once the CSS is enabled and if the HSE clock fails, the CSSI occurs and a NMI is automatically generated. The NMI is executed indefinitely unless the CSSI pending bit is cleared. As a consequence, in the NMI ISR, the user must clear the CSSI by setting the CSSC bit in the RCC 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 MSI or the HSI16 oscillator depending on the STOPWUCK configuration in the *RCC clock configuration register (RCC_CFGR)*, 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.

9.3.12 Clock security system on LSE

A clock security system on LSE can be activated by software writing the LSECSSON bit in the *RCC Backup domain control register (RCC_BDCR)*. This bit can be disabled only by a hardware reset or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE and LSI are enabled (LSEON and LSION enabled) and ready (LSERDY and LSIRDY set by hardware, LSIPRE disabled), 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 MSI was in PLL-mode, this mode is disabled.

The CSS on LSE failure is detected by a tamper event.

In Standby mode a wakeup is generated. In other modes a TAMP interrupt can be sent to wake up the software (see *Table 310: TAMP interconnection* and *Section 42.5: TAMP interrupts*).

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.

9.3.13 ADC clock

The ADC clock is derived from the system clock, or from the PLLSAI1 or the PLLSAI2 output. It can reach 110 MHz and can be divided by the following prescalers values: 1,2,4,6,8,10,12,16,32,64,128 or 256 by configuring the ADC1_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 ADC123_CCR register.

If the programmed factor is 1, the AHB prescaler must be set to 1.

9.3.14 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 *RCC Backup domain control register (RCC_BDCR)*. This selection cannot be modified without resetting the Backup domain. The system must always be configured so as to get a PCLK frequency greater then or equal to the RTCCLK frequency for a proper operation of the RTC.

The LSE clock is in the Backup domain, whereas the 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.

9.3.15 Timer clock

The timer clock frequencies are automatically defined by hardware.

There are two cases:

- If the APB prescaler equals 1, the timer clock frequencies are set to the APB domain frequency.
- Otherwise, they are set to twice (×2) the APB domain frequency.

9.3.16 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 LSI 32 kHz clock is provided to the IWDG.

9.3.17 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 MCO clock.

  – LSI
  – LSE
  – SYSCLK
  – HSI16
  – HSI48
  – HSE
  – PLLCLK
  – MSI

  The selection is controlled by the MCOSEL[3:0] bits of the *RCC clock configuration register (RCC_CFGR)*. The selected clock can be divided with the MCOPRE[2:0] field of the *RCC clock configuration register (RCC_CFGR)*.

- **LSCO**

  Another output (LSCO) allows one of the low-speed clocks below to be output onto the external LSCO pin:

  – LSI
  – LSE

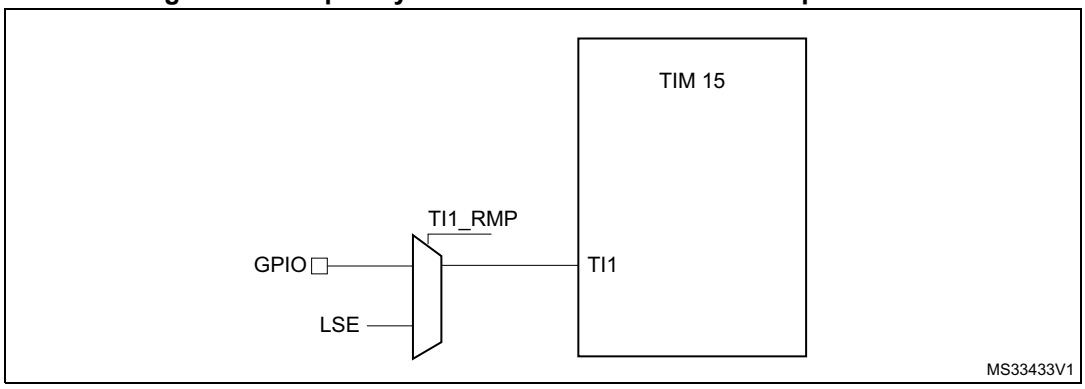
  This output remains available in Stop (Stop 0, Stop 1 and Stop 2) and Standby modes. The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN in the *RCC Backup domain control register (RCC_BDCR)*.

The MCO clock output requires the corresponding alternate function selected on the MCO pin. The LSCO pin must be left in default POR state.

9.3.18 Internal/external clock measurement with TIM15/TIM16/TIM17

It is possible to indirectly measure the frequency of all on-board clock sources by mean of the TIM15, TIM16 or TIM17 channel 1 input capture, as represented *Figure 32*, *Figure 33* and *Figure 34*.

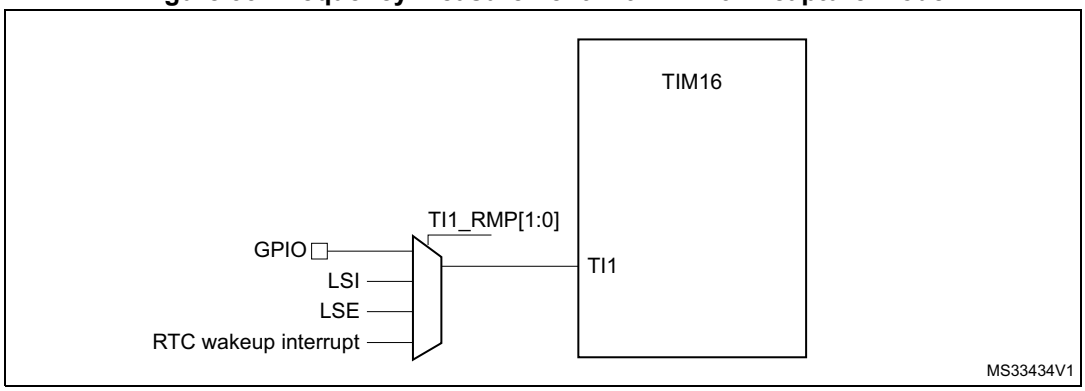
Figure 32. Frequency measurement with TIM15 in capture mode



The input capture channel of the Timer 15 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1_RMP bit in the TIM15_OR register. 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.

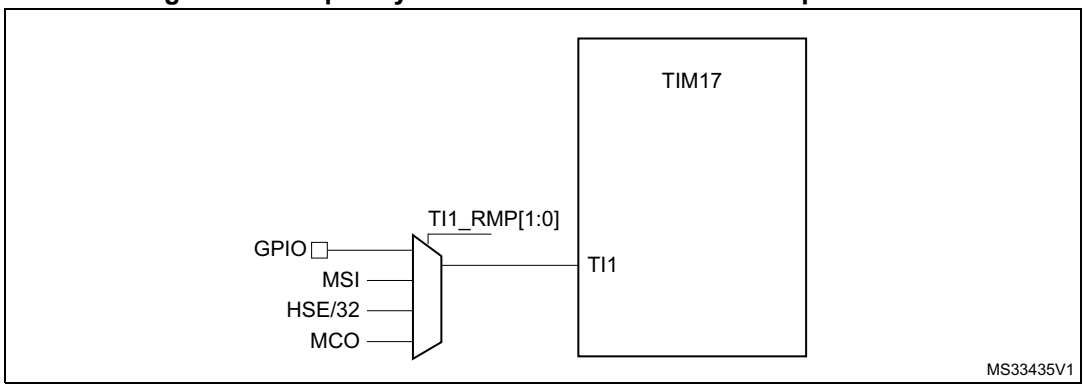
Figure 33. Frequency measurement with TIM16 in capture mode



The input capture channel of the Timer 16 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1_RMP[1:0] bits in the TIM16_OR register. 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.

Figure 34. 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. This selection is performed through the TI1_RMP[1:0] bits in the TIM17_OR register. 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 MSI Clock.
- 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 RCC_CFGR register.

Calibration of the HSI16 and the MSI

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 and MSI system clocks (for this, either HSI16 or MSI must be used as system clock source). The number of HSI16 (MSI, respectively) clock counts between consecutive edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppms), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate manufacturing, process, temperature and/or voltage related frequency deviations.

The MSI and HSI16 oscillator both have dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (such as 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.

It is however not possible to have a good enough resolution when the MSI clock is low (typically below 1 MHz). In this case, the following steps are needed:

1. Accumulate the results of several captures in a row.
2. Use the timer input capture prescaler (up to 1 capture every 8 periods).
3. Use the RTC wakeup interrupt signal (when the RTC is clocked by the LSE) as the input for the channel1 input capture. This improves the measurement precision. For this purpose the RTC wakeup interrupt must be enabled.

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 defining the HSE as system clock source, the number of its 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 (such as 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.

9.3.19 Peripheral clock enable registers (RCC_AHBxENR, RCC_APBxENRy)

Each peripheral clock can be enabled by the corresponding EN bit in the RCC_AHBxENR and RCC_APBxENRy registers.

When the peripheral clock is not active, read or write accesses to the peripheral registers 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 is active.

Caution: Just after enabling the clock for a peripheral, the software **must** wait for a delay before accessing the peripheral registers.

9.4 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and Low-power sleep modes stop the CPU clock. The memory interface clocks (Flash memory, SRAM1 and SRAM2 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, Stop 1 and Stop 2) stop all the clocks in the V_{CORE} domain and disable the three PLL, HSI16, MSI and HSE oscillators.

  All U(S)ARTs, LPUARTs and I2Cs have the capability to enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as 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 (no capability to turn on the LSE oscillator).
- Standby and Shutdown modes stop all the clocks in the V_{CORE} domain and disable the PLLs, HSI16, MSI and 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 exiting Stop modes (Stop 0, Stop 1 or Stop 2), the system clock is either MSI or HSI16, depending on the software configuration of the STOPWUCK bit in the RCC_CFGR register. The frequency (range and user trim) of the MSI oscillator is the one configured

before entering Stop mode. The user trim of HSI16 is kept. If the MSI was in PLL-mode before entering Stop mode, the PLL-mode stabilization time must be waited for after wakeup even if the LSE was kept ON during the Stop mode.

When leaving the Standby and Shutdown modes, the system clock is MSI. The MSI frequency at wakeup from Standby mode is configured with the MSISRANGE is the RCC_CSR register, from 1 to 8 MHz. The MSI frequency at wakeup from Shutdown mode is 4 MHz. The user trim is lost.

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.

9.5 RCC TrustZone® security

When the TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register, the RCC is able to secure RCC configuration and status bits from being modified by non-secure accesses.

This is configured through the security configuration register RCC_SECCFGR to prevent non-secure access to read or modify the following features:

- HSE, HSE-CSS, HSI, MSI, LSI, LSE, LSE-CSS, HSI48 configuration and status bits
- main PLL, PLLSAI1, PLLSAI2, AHB prescaler configuration and status bits
- system clock SYSCLK and HSI48 source clock selection and status bits
- MCO clock output configuration and STOPWUCK bit
- reset flag RMVF configuration

When a peripheral is configured as secure, its related clock, reset, clock source selection and clock enable during low-power modes control bits are also secure in the RCC_AHBxENR, RCC_APBxENR, RCC_AHBxSMEN, RCC_APBxSMEN, RCC_CCIPR1 and RCC_CCIPR2 registers.

A peripheral is secure when:

- For securable peripherals by TZSC (TrustZone security controller), the SEC security bit corresponding to this peripheral is set in the TZSC_SECCFGRx register.
- For TrustZone-aware peripherals, a security feature of this peripheral is enabled through its dedicated bits.

Table 78 gives a summary of the RCC secured bits following the security configuration bit in the RCC_SECCFGR register.

When one security configuration bit is set, some configuration and status bits are secured. The RCC registers may contain secure and non-secure bits:

- Secured bits: read and write operations are only allowed by a secure access. Non-secure read or write accesses are RAZ/WI. There is no illegal access event generated.
- Non-secure bits: no restriction. Read and write operations are allowed by both secure and non-secure accesses.
- A non-secure read/write access to RCC_SECCFGR register is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the RCC illegal access interrupt is enabled in the TZIC_IER2 register.

When the TrustZone security is disabled (TZEN = 0 in FLASH_OPTR register), all registers are non-secure. The RCC_SECCFGR secure register and security status registers are RAZ/WI.

Table 78. RCC security configuration summary

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
HSISEC	HSION, HSIKERON, HSIRDY, HSIASFS	RCC_CR
	HSICAL[7:0], HSITRIM[6:0]	RCC_ICSCR
	HSIRDYIE	RCC_CIER
	HSIRDYC	RCC_CICR
HSESEC	HSEON, HSERDY, HSEBYP, HSECSSON	RCC_CR
	HSERDYIE	RCC_CIER
	HSERDYC, HSECSSC	RCC_CICR
MSISEC	MSION, MSIRDY, MSIPLLEN, MSIRGSEL, MSISRANGE[3:0]	RCC_CR
	MSICAL[7:0], MSITRIM[7:0]	RCC_ICSCR
	MISIRANGE[3:0]	RCC_CSR
	MSIRDYIE	RCC_CIER
LSISEC	LSIO, LSIRDY, LSIPRE	RCC_CSR
	LSIRDYIE	RCC_CIER
	LSIRDYC	RCC_CICR
LSESEC	LSECSSON, LSECSSD, LSEDRV[1:0], LSEBYP, LSERDY, LSEON, LSCOSEL, LSCOEN	RCC_BDCR
	LSERDYIE, LSECSSIE	RCC_CIER
	LSERDYC, LSECSSC	RCC_CICR
SYSCLKSEC	SW[1:0], SWS[1:0], STOPWUCK, MCOSEL[3:0], MCOPRE[2:0],	RCC_CFGR
	VOS[1:0]	PWR_CR1
PRESCSEC	HPRE[3:0], PPRE1[2:0], PPRE2[2:0]	RCC_CFGR
PLLSEC	PLLSRC[1:0], PLLM[3:0], PLLN[6:0], PLLPDIV[4:0], PLLR[1:0], PLLREN, PLLQ[1:0], PLLP, PLLPEN, PLLQEN	RCC_PLLCFGR
	PLLRDY, PLLON	RCC_CR
	PLLRDYIE	RCC_CIER
	PLLRDYC	RCC_CICR
PLLSAI1SEC	PLLSAI1PDIV[4:0], PLLSAI1R[1:0], PLLSAI1REN, PLLSAI1Q[1:0], PLLSAI1QEN, PLLSAI1P, PLLSAI1PEN, PLLSAI1N[6:0], PLLSAI1M[3:0], PLLSAI1SRC[1:0]	RCC_PLLSAI1CFGR
	PLLSAI1RDY, PLLSAI1ON	RCC_CR
	PLLSAI1RDYIE	RCC_CIER
	PLLSAI1RDYC	RCC_CICR

Table 78. RCC security configuration summary (continued)

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
PLLSAI2SEC	PLLSAI2PDIV[4:0], PLLSAI2P, PLLSAI2PEN, PLLSAI2N[6:0], PLLSAI2M[3:0], PLLSAI2SRC[1:0]	RCC_PLLSAI2CFGR
	PLLSAI2RDY, PLLSAI2ON	RCC_CR
	PLLSAI2RDYIE	RCC_CIER
	PLLSAI2RDYC	RCC_CICR
HSI48SEC	HSI48CAL[8:0], HSI48RDY	RCC_CRRCR
	HSI48RDYE	RCC_CIER
	HSI48RDYC	RCC_CICR
CLK48MSEC	CLK48MSEL[1:0]	RCC_CCIPR1
RMVFSEC	RMVF	RCC_CSR

9.6 RCC Privileged and Unprivileged mode

By default, after reset, all RCC registers can be read or written in both Privileged and Unprivileged modes except RCC privilege bit (PRIV bit in RCC_CR) that can be written in Privilege mode only.

When the PRIV bit is set in RCC_CR register:

- Writing the RCC registers is possible only in privileged mode.
- All RCC registers can be read only in privileged mode except RCC security status registers (RCC_AHBxSECSR, RCC_APBx_SECSR, RCC_SECSR) and PRIV bit in RCC_CR register.
- An unprivileged access to a privileged RCC register is discarded. RAZ/WI.

9.7 RCC interrupts

The RCC provides three interrupt lines:

- **rcc_it**: general interrupt line providing events when the PLLs are ready or when the oscillators are ready
- **rcc_hsecss_it**: interrupt line dedicated to the failure detection of the HSE CSS (clock security system)
- **rcc_lsecss_it**: interrupt line dedicated to the failure detection of the LSE CSS

The interrupt enable is controlled via *RCC clock interrupt enable register (RCC_CIER)*, except for the HSE CSS failure. When the HSE CSS feature is enabled, it not possible to mask the interrupt generation.

The interrupt flags can be checked via *RCC clock interrupt flag register (RCC_CIFR)*, and those flags can be cleared via *RCC clock interrupt clear register (RCC_CICR)*.

Note: The interrupt flags are not relevant if the corresponding interrupt enable bit is not set.

Table 79 gives a summary of the interrupt sources and the way to control them.

Table 79. Interrupt sources and control(1)

Interrupt vector	Interrupt event flag	Description	Enable control bit	Interrupt clear method	Interrupt internal signal
RCC RCC_S	LSIRDYF	LSI ready	LSIRDYIE	Set LSIRDYC to 1	rcc_it
	LSERDYF	LSE ready	LSERDYIE	Set LSERDYC to 1	
	HSIDRYF	HSI ready	HSIDRYIE	Set HSIRDYC to 1	
	HSERDYF	HSE ready	HSERDYIE	Set HSERDYC to 1	
	CSIRDYF	CSI ready	CSIRDYIE	Set CSIRDYC to 1	
	HSI48RDYF	HSI48 ready	HSI48RDYIE	Set HSI48RDYC to 1	
	PLLRDYF	PLL ready	PLLRDYIE	Set PLLRDYC to 1	
	PLLSAI1RDYF	PLLSAI1 ready	PLLSAI1RDYIE	Set PLLSAI1RDYC to 1	
	PLLSAI2RDYF	PLLSAI2 ready	PLLSAI2RDYIE	Set PLLSAI2RDYC to 1	
TAMP TAMP_S	ITAMP3F	LSE CSS failure	ITAMP3IE (2)	Set CITAMP3F to 1	rcc_lsecss_it
NMI	HSECSSF	HSE CSS failure	-(3)	Set HSECSSC to 1	rcc_hsecss_it

1. When TrustZone is enabled, two interrupt vectors are available for secure and non-secure events.
2. The security system feature must also be enabled (LSECSSON = 1), in order to generate interrupts.
3. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).

9.8 RCC registers

9.8.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0063

HSEBYP is cleared upon power-on reset. It is not affected upon other types of 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
PRIV	Res.	PLLSAI2RDY	PLLSAI2ON	PLLSAI1RDY	PLLSAI1ON	PLLRDY	PLLON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON
rw		r	rw	r	rw	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.	HSIASFS	HSIRDY	HSIKERON	HSION	MSIRANGE[3:0]				MSIRGSEL	MSIPLLEN	MSIRDY	MSION
				rw	r	rw	rw	rw	rw	rw	rw	rs	rw	r	rw

Bit 31 **PRIV**: RCC privilege

Set and reset by software. This bit can be read by both privileged or unprivileged access. when set, it can only be cleared by a privileged access.

0: RCC registers can be accessed by a privileged or non-privileged access.

1: RCC registers can be accessed only by a privileged access except RCC_AHBxSECSR, RCC_APBx_SECSR and RCC_SECSR.

An unprivileged access to RCC registers is RAZ/WI.

If TrustZone security is enabled (TZEN = 1), when the RCC is not secure, the PRIV bit can be written by a secure or non-secure privileged access.

If the RCC is secure, the PRIV bit can be written only by a secure privileged access. A non-secure write access is ignored.

A secure unprivileged write access on PRIV bit is ignored.

Bit 30 Reserved, must be kept at reset value.

Bit 29 **PLLSAI2RDY**: SAI2 PLL clock ready flag

Set by hardware to indicate that the PLLSAI2 is locked.

0: PLLSAI2 unlocked

1: PLLSAI2 locked

Bit 28 **PLLSAI2ON**: SAI2 PLL enable

Set and cleared by software to enable PLLSAI2.

Cleared by hardware when entering Stop, Standby or Shutdown mode.

0: PLLSAI2 OFF

1: PLLSAI2 ON

Bit 27 **PLLSAI1RDY**: SAI1 PLL clock ready flag

Set by hardware to indicate that the PLLSAI1 is locked.

0: PLLSAI1 unlocked

1: PLLSAI1 locked

Bit 26 **PLLSAI1ON**: SAI1 PLL enable
Set and cleared by software to enable PLLSAI1.
Cleared by hardware when entering Stop, Standby or Shutdown mode.
0: PLLSAI1 OFF
1: PLLSAI1 ON

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 15:12 Reserved, must be kept at reset value.

Bit 11 **HSIASFS**: HSI16 automatic start from Stop
Set and cleared by software. When the system wakeup clock is MSI, this bit is used to wakeup the HSI16 is parallel of the system wakeup.
0: HSI16 oscillator is not enabled by hardware when exiting Stop mode with MSI as wakeup clock.
1: HSI16 oscillator is enabled by hardware when exiting Stop mode with MSI as wakeup clock.

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^2Cs 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 7:4 **MSIRANGE[3:0]**: MSI clock ranges

These bits are configured by software to choose the frequency range of MSI when MSIRGSEL is set.12 frequency ranges are available:

0000: range 0 around 100 kHz

0001: range 1 around 200 kHz

0010: range 2 around 400 kHz

0011: range 3 around 800 kHz

0100: range 4 around 1M Hz

0101: range 5 around 2 MHz

0110: range 6 around 4 MHz (reset value)

0111: range 7 around 8 MHz

1000: range 8 around 16 MHz

1001: range 9 around 24 MHz

1010: range 10 around 32 MHz

1011: range 11 around 48 MHz

others: not allowed (hardware write protection)

Note: Warning: MSIRANGE can be modified when MSI is OFF (MSION=0) or when MSI is ready (MSIRDY=1). MSIRANGE must NOT be modified when MSI is ON and NOT ready (MSION=1 and MSIRDY=0)

Bit 3 **MSIRGSEL**: MSI clock range selection

Set by software to select the MSI clock range with MSIRANGE[3:0]. Write 0 has no effect. After a standby or a reset MSIRGSEL is at 0 and the MSI range value is provided by MSISRANGE in CSR register.

0: MSI Range is provided by MSISRANGE[3:0] in RCC_CSR register

1: MSI Range is provided by MSIRANGE[3:0] in the RCC_CR register

Bit 2 **MSIPLLEN**: MSI clock PLL enable

Set and cleared by software to enable/ disable the PLL part of the MSI clock source.
MSIPLLEN must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware).There is a hardware protection to avoid enabling MSIPLLEN if LSE is not ready.
This bit is cleared by hardware when LSE is disabled (LSEON = 0) or when the Clock Security System on LSE detects a LSE failure (refer to RCC_CSR register).
0: MSI PLL OFF
1: MSI PLL ON

Bit 1 **MSIRDY**: MSI clock ready flag

This bit is set by hardware to indicate that the MSI oscillator is stable.
0: MSI oscillator not ready
1: MSI oscillator ready

Note: Once the MSION bit is cleared, MSIRDY goes low after 6 MSI clock cycles.

Bit 0 **MSION**: MSI clock enable

This bit is set and cleared by software.
Cleared by hardware to stop the MSI oscillator when entering Stop, Standby or Shutdown mode.
Set by hardware to force the MSI oscillator ON when exiting Standby or Shutdown mode.
Set by hardware to force the MSI oscillator ON when STOPWUCK=0 when exiting from Stop modes, or in case of a failure of the HSE oscillator
Set by hardware when used directly or indirectly as system clock.
0: MSI oscillator OFF
1: MSI oscillator ON

9.8.2 RCC internal clock sources calibration register (RCC_ICSCR)

Address offset: 0x004

Reset value: 0x40XX 00XX

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
MSITRIM[7:0]								MSICAL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSITRIM[6:0]**: HSI16 clock trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI16.

Bits 23:16 **HSICAL[7:0]**: HSI16 clock calibration

These bits are initialized at startup with the factory-programmed HSI16 calibration trim value. When HSITRIM is written, HSICAL is updated with the sum of HSITRIM and the factory trim value.

Bits 15:8 **MSITRIM[7:0]**: MSI clock trimming

These bits provide an additional user-programmable trimming value that is added to the MSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the MSI.

Bits 7:0 **MSICAL[7:0]**: MSI clock calibration

These bits are initialized at startup with the factory-programmed MSI calibration trim value. When MSITRIM is written, MSICAL is updated with the sum of MSITRIM and the factory trim value.

9.8.3 RCC clock configuration register (RCC_CFGR)

Address offset: 0x008

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states are inserted only if the access occurs during clock source switch.

From 0 to 15 wait states are 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
STOPWUCK	Res.	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **MCOPRE[2:0]**: microcontroller clock output prescaler

These bits are set and cleared by software.
It is highly recommended to change this prescaler before MCO output is enabled.
000: MCO divided by 1
001: MCO divided by 2
010: MCO divided by 4
011: MCO divided by 8
100: MCO divided by 16
Others: not allowed

Bits 27:24 **MCOSEL[3:0]**: microcontroller clock output
Set and cleared by software.
0000: MCO output disabled, no clock on MCO
0001: SYSCLK system clock selected
0010: MSI clock selected
0011: HSI16 clock selected
0100: 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 23:16 Reserved, must be kept at reset value.

Bit 15 **STOPWUCK:** wakeup from Stop and CSS backup clock selection
Set and cleared by software to select the system clock used when exiting Stop mode.
The selected clock is also used as emergency clock for the clock security system on HSE.
Warning: STOPWUCK must not be modified when the CSS is enabled by HSECSSON bit in the RCC_CR register and the system clock is HSE (SWS=”10”) or a switch on HSE is requested (SW=”10”).
0: MSI oscillator selected as wakeup from stop clock and CSS backup clock
1: HSI16 oscillator selected as wakeup from stop clock and CSS backup clock

Bit 14 Reserved, must be kept at reset value.

Bits 13:11 **PPRE2[2:0]**: APB high-speed prescaler (APB2)
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]:**APB low-speed prescaler (APB1)
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 must set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details, refer to *Section 8.2.5: 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: MSI oscillator used as system clock
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 MSI oscillator selection when exiting Standby or Shutdown mode. Configured by hardware to force MSI or HSI16 oscillator selection when exiting Stop mode or in case of failure of the HSE oscillator, depending on STOPWUCK value.
00: MSI selected as system clock
01: HSI16 selected as system clock
10: HSE selected as system clock
11: PLL selected as system clock

9.8.4 RCC PLL configuration register (RCC_PLLCFGR)

Address offset: 0x00C

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]		PLLREN	Res.	PLLQ[1:0]		PLLQEN	Res.	Res.	PLLP	PLLPEN
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 PLL division factor for PLLSAI3CLK

Set and cleared by software to control the SAI1 or SAI2 clock frequency.
PLLSAI3CLK output clock frequency = VCO frequency / PLLPDIV.
00000: PLLSAI3CLK is controlled by the bit PLLP
00001: Reserved
00010: PLLSAI3CLK = VCO / 2
....
11111: PLLSAI3CLK = VCO / 31

Bits 26:25 **PLLR[1:0]**: Main PLL division factor for PLLCLK (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.
PLLCLK 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 must set these bits correctly not to exceed 110 MHz on this domain.

Bit 24 **PLLREN**: Main PLL PLLCLK output enable

Set and reset by software to enable the PLLCLK output of the main PLL (used as system clock).
This bit cannot be written when PLLCLK output of the PLL is used as system clock.
In order to save power, when the PLLCLK output of the PLL is not used, the value of PLLREN must be 0.
0: PLLCLK output disabled
1: PLLCLK output enabled

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **PLLQ[1:0]**: Main PLL division factor for PLL48M1CLK (48 MHz clock)

Set and cleared by software to control the frequency of the main PLL output clock PLL48M1CLK. This output can be selected for USB, RNG, SDMMC (48 MHz clock). These bits can be written only if PLL is disabled.

PLL48M1CLK 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 must set these bits correctly not to exceed 80 MHz on this domain.

Bit 20 **PLLQEN**: Main PLL PLL48M1CLK output enable

Set and reset by software to enable the PLL48M1CLK output of the main PLL.

In order to save power, when the PLL48M1CLK output of the PLL is not used, the value of PLLQEN must be 0.

0: PLL48M1CLK output disabled

1: PLL48M1CLK output enabled

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **PLLP**: Main PLL division factor for PLLSAI3CLK (SAI1 and SAI2 clock)

Set and cleared by software to control the frequency of the main PLL output clock PLLSAI3CLK. This output can be selected for SAI1 or SAI2. These bits can be written only if PLL is disabled.

When the PLLPDIV[4:0] is set to 0x0, PLLSAI3CLK output clock frequency = VCO frequency / PLLP with PLLP =7, or 17.

0: PLLP = 7

1: PLLP = 17

Caution: The software must set these bits correctly not to exceed 80 MHz on this domain.

Bit 16 **PLLPEN**: Main PLL PLLSAI3CLK output enable

Set and reset by software to enable the PLLSAI3CLK output of the main PLL.

In order to save power, when the PLLSAI3CLK output of the PLL is not used, the value of PLLPEN must be 0.

0: PLLSAI3CLK output disabled

1: PLLSAI3CLK output enabled

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 * PLLN with 8 ≤ PLLN ≤ 86
0000000: PLLN = 0 wrong configuration
0000001: PLLN = 1 wrong configuration
...
0000111: PLLN = 7 wrong configuration
0001000: PLLN = 8
0001001: PLLN = 9
...
1111111: PLLN = 127 wrong configuration

Caution: The software must set correctly these bits to secure that the VCO output frequency is between 64 and 344 MHz.

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 ≤ 8
0000: PLLM = 1
0001: PLLM = 2
0010: PLLM = 3
0011: PLLM = 4
0100: PLLM = 5
0101: PLLM = 6
0110: PLLM = 7
0111: PLLM = 8
...
1111: PLLM = 16

Caution: The software must set these bits correctly to ensure that the VCO input frequency is between 4 to 16 MHz.

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 the PLL is disabled.
In order to save power, when no PLL is used, the value of PLLSRC must be 00.
00: No clock sent to PLL
01: MSI clock selected as PLL clock entry
10: HSI16 clock selected as PLL clock entry
11: HSE clock selected as PLL clock entry

9.8.5 RCC PLLSAI1 configuration register (RCC_PLLSAI1CFGR)

Address offset: 0x010

Reset value: 0x0000 1000

Access: no wait state, word, half-word and byte access

This register is used to configure the PLLSAI1 clock outputs according to the formulas:

- f(VCOSAI1 clock) = f(PLL clock input) * (PLLSAI1N / PLLM)
- f(PLLSAI1_P) = f(VCOSAI1 clock) / PLLSAI1PDIV
- f(PLLSAI1_Q) = f(VCOSAI1 clock) / PLLSAI1Q
- f(PLLSAI1_R) = f(VCOSAI1 clock) / PLLSAI1R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLSAI1PDIV[4:0]					PLLSAI1R[1:0]		PLLSAI1REN	Res.	PLLSAI1Q[1:0]		PLLSAI1QEN	Res.	Res.	PLLSAI1P	PLLSAI1PEN
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.	PLLSAI1N[6:0]							PLLSAI1M[3:0]				Res.	Res.	PLLSAI1SRC[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:27 **PLLSAI1PDIV[4:0]**: PLLSAI1 division factor for PLLSAI1CLK
Set and cleared by software to control the SAI1 or SAI2 clock frequency.
PLLSAI1CLK output clock frequency = VCOSAI1 frequency / PLLSAI1PDIV.
00000: PLLSAI1CLK controlled by PLLSAI1P bit
00001: reserved
00010: PLLSAI1CLK = VCOSAI1 / 2
....
11111: PLLSAI1CLK = VCOSAI1 / 31

Note: This bit can be written only when the PLLSAI1 is disabled.

Bits 26:25 **PLLSAI1R[1:0]**: PLLSAI1 division factor for PLLADC1CLK (ADC clock)
Set and cleared by software to control the frequency of the PLLSAI1 output clock PLLADC1CLK. This output can be selected as ADC clock.
These bits can be written only if PLLSAI1 is disabled.
PLLADC1CLK output clock frequency = VCOSAI1 frequency / PLLSAI1R, with PLLSAI1R = 2, 4, 6, or 8.
00: PLLSAI1R = 2
01: PLLSAI1R = 4
10: PLLSAI1R = 6
11: PLLSAI1R = 8

Bit 24 **PLLSAI1REN**: PLLSAI1 PLLADC1CLK output enable
Set and reset by software to enable the PLLADC1CLK output of the PLLSAI1 (used as clock for ADC).
In order to save power, when the PLLADC1CLK output of the PLLSAI1 is not used, the value of PLLSAI1REN must be 0.
0: PLLADC1CLK output disabled
1: PLLADC1CLK output enabled

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **PLLSAI1Q[1:0]**: PLLSAI1 division factor for PLL48M2CLK (48 MHz clock)
Set and cleared by software to control the frequency of the PLLSAI1 output clock PLL48M2CLK. This output can be selected for USB, RNG, SDMMC (48 MHz clock). These bits can be written only if PLLSAI1 is disabled.
PLL48M2CLK output clock frequency = VCOSAI1 frequency / PLLSAI1Q with PLLSAI1Q = 2, 4, 6, or 8
00: PLLSAI1Q = 2
01: PLLSAI1Q = 4
10: PLLSAI1Q = 6
11: PLLSAI1Q = 8

Caution: The software must set these bits correctly not to exceed 110 MHz on this domain.

Bit 20 **PLLSAI1QEN**: PLLSAI1 PLL48M2CLK output enable
Set and reset by software to enable the PLL48M2CLK output of the PLLSAI1.
In order to save power, when the PLL48M2CLK output of the PLLSAI1 is not used, the value of PLLSAI1QEN must be 0.
0: PLL48M2CLK output disablde
1: PLL48M2CLK output enabled

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **PLLSAI1P**: PLLSAI1 division factor for PLLSAI1CLK (SAI1 or SAI2 clock).
Set and cleared by software to control the frequency of the PLLSAI1 output clock PLLSAI1CLK. This output can be selected for SAI1 or SAI2. These bits can be written only if PLLSAI1 is disabled.
When the PLLSAI1PDIV[4:0] is set to 0x0, PLLSAI1CLK output clock frequency = VCOSAI1 frequency / PLLSAI1P with PLLSAI1P =7, or 17
0: PLLSAI1P = 7
1: PLLSAI1P = 17

Bit 16 **PLLSAI1PEN**: PLLSAI1 PLLSAI1CLK output enable
Set and reset by software to enable the PLLSAI1CLK output of the PLLSAI1.
In order to save power, when the PLLSAI1CLK output of the PLLSAI1 is not used, the value of PLLSAI1PEN must be 0.
0: PLLSAI1CLK output disabled
1: PLLSAI1CLK output enabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLSAI1N[6:0]**: PLLSAI1 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 PLLSAI1 is disabled.
VCOSAI1 output frequency = VCOSAI1 input frequency x PLLSAI1N, with
8 ≤ PLLSAI1N ≤ 86.
0000000: PLLSAI1N = 0 wrong configuration
0000001: PLLSAI1N = 1 wrong configuration
...
0000111: PLLSAI1N = 7 wrong configuration
0001000: PLLSAI1N = 8
0001001: PLLSAI1N = 9
...
1111111: PLLSAI1N = 127

Caution: The software must set correctly these bits to ensure that the VCO output frequency is between 64 and 344 MHz.

Bits 7:4 **PLLSAI1M[3:0]**: Division factor for PLLSAI1 input clock

Set and reset by software to divide the PLLSAI1 input clock before the VCO.
These bits can be written only when PLLSAI1 is disabled.
VCO input frequency = PLLSAI1 input clock frequency / PLLSAI1M with 1 ≤ PLLSAI1 ≤ 16.
0000: PLLSAI1M = 1
0001: PLLSAI1M = 2
0010: PLLSAI1M = 3
0011: PLLSAI1M = 4
0100: PLLSAI1M = 5
0101: PLLSAI1M = 6
0110: PLLSAI1M = 7
0111: PLLSAI1M = 8
1000: PLLSAI1M = 9
...
1111: PLLSAI1M= 16

Caution: The software must set these bits correctly to ensure that the VCO input frequency ranges from 2.66 to 8 MHz.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSAI1SRC[1:0]**: Main PLLSAI1 entry clock source

Set and cleared by software to select PLLSAI1 clock source.
These bits can be written only when PLLSAI1 is disabled.
In order to save power, when PLLSAI1 is not used, the value of PLLSAI1 must be 00.
00: No clock sent to PLLSAI1
01: MSI clock selected as PLLSAI1 clock entry
10: HSI16 clock selected as PLLSAI1 clock entry
11: HSE clock selected as PLLSAI1 clock entry

9.8.6 RCC PLLSAI2 configuration register (RCC_PLLSAI2CFGR)

Address offset: 0x014

Reset value: 0x0000 1000

Access: no wait state, word, half-word and byte access

This register is used to configure the PLLSAI2 clock outputs according to the formulas:

- f(VCOSAI2 clock) = f(PLL clock input) × (PLLSAI2N / PLLM)
- f(PLLSAI2_P) = f(VCOSAI2 clock) / PLLSAI2P
- f(PLLSAI2_R) = f(VCOSAI2 clock) / PLLSAI2R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLSAI2PDIV[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLSAI2P	PLLSAI2PEN
rw	rw	rw	rw	rw										rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PLLSAI2N[6:0]							PLLSAI2M[3:0]				Res.	Res.	PLLSAI2SRC[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:27 **PLLSAI2PDIV[4:0]**: PLLSAI2 division factor for PLLSAI2CLK
Set and cleared by software to control the SAI1 or SAI2 clock frequency.
PLLSAI2CLK output clock frequency = VCOSAI2 frequency / PLLSAI2PDIV.
00000: PLLSAI2CLK controlled by the bit PLLSAI2P
00001: reserved
00010: PLLSAI2CLK = VCOSAI2 / 2
....
11111: PLLSAI2CLK = VCOSAI2 / 31

Bits 26:18 Reserved, must be kept at reset value.

Bit 17 **PLLSAI2P**: PLLSAI2 division factor for PLLSAI2CLK (SAI1 or SAI2 clock).
Set and cleared by software to control the frequency of the PLLSAI2 output clock PLLSAI2CLK. This output can be selected for SAI1 or SAI2.
These bits can be written only if PLLSAI2 is disabled.
When the PLLSAI2PDIV[4:0] is set to 0x0, PLLSAI2CLK output clock frequency = VCOSAI2 frequency / PLLSAI2P with PLLSAI2P =7 or 17.
0: PLLSAI2P = 7
1: PLLSAI2P = 17

Bit 16 **PLLSAI2PEN**: PLLSAI2 PLLSAI2CLK output enable
Set and reset by software to enable the PLLSAI2CLK output of the PLLSAI2.
In order to save power, when the PLLSAI2CLK output of the PLLSAI2 is not used, the value of PLLSAI2PEN must be 0.
0: PLLSAI2CLK output disabled
1: PLLSAI2CLK output enabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLSAI2N[6:0]**: PLLSAI2 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 PLLSAI2 is disabled.
VCOSAI2 output frequency = VCOSAI2 input frequency x PLLSAI2N, with
8 ≤ PLLSAI2N ≤ 86
0000000: PLLSAI2N = 0 wrong configuration
0000001: PLLSAI2N = 1 wrong configuration
...
0000111: PLLSAI2N = 7 wrong configuration
0001000: PLLSAI2N = 8
0001001: PLLSAI2N = 9
...
1111111: PLLSAI2N = 127 wrong configuration

Caution: The software must set correctly these bits to ensure that the VCO output frequency is between 64 and 344 MHz.

Bits 7:4 **PLLSAI2M[3:0]**: Division factor for PLLSAI2 input clock

Set and reset by software to divide the PLLSAI2 input clock before the VCO.
These bits can be written only when PLLSAI2 is disabled.
VCO input frequency = PLLSAI2 input clock frequency / PLLM with 1 ≤ PLLSAI2N ≤ 16
0000: PLLSAI2M = 1
0001: PLLSAI2M = 2
0010: PLLSAI2M = 3
0011: PLLSAI1M = 4
0100: PLLSAI2M = 5
0101: PLLSAI2M = 6
0110: PLLSAI2M = 7
0111: PLLSAI2M = 8
1000: PLLSAI2M = 9
...
1111: PLLSAI2M= 16

Caution: The software must set these bits correctly to ensure that the VCO input frequency is between 2.66 to 8 MHz.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSAI2SRC[1:0]**: Main PLLSAI2 entry clock source

Set and cleared by software to select PLLSAI2 clock source.
These bits can be written only when PLLSAI2 is disabled.
In order to save power, when PLLSAI2 is not used, the value of PLLSAI2 must be 00.
00: No clock sent to PLLSAI2
01: MSI clock selected as PLLSAI2 clock entry
10: HSI16 clock selected as PLLSAI2 clock entry
11: HSE clock selected as PLLSAI2 clock entry

9.8.7 RCC clock interrupt enable register (RCC_CIER)

Address offset: 0x018

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSI48RDYIE	Res.	Res.	PLLSAI2RDYIE	PLLSAI1RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	MSIRDYIE	LSERDYIE	LSIRDYIE
					rw			rw	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 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **PLLSAI2RDYIE**: PLLSAI2 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLLSAI2 lock.
0: PLLSAI2 lock interrupt disabled
1: PLLSAI2 lock interrupt enabled

Bit 6 **PLLSAI1RDYIE**: PLLSAI1 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLSAI1L lock.
0: PLLSAI1 lock interrupt disabled
1: PLLSAI1 lock interrupt enabled

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 **MSIRDYIE**: MSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the MSI oscillator stabilization.
0: MSI ready interrupt disabled
1: MSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
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

9.8.8 RCC clock interrupt flag register (RCC_CIFR)

Address offset: 0x01C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSI48RDYF	Res.	CSSF	PLLSAI2RDYF	PLLSAI1RDYF	PLLRDYF	HSERDYF	HSIRDYF	MSIRDYF	LSERDYF	LSIRDYF
					r		r	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 *RCC 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 Reserved, must be kept at reset value.

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

Bit 7 **PLLSAI2RDYF**: PLLSAI2 ready interrupt flag
Set by hardware when the PLLSAI2 locks and PLLSAI2RDYDIE is set.
Cleared by software setting the PLLSAI2RDYC bit.
0: No clock ready interrupt caused by PLLSAI2 lock
1: Clock ready interrupt caused by PLLSAI2 lock

Bit 6 **PLLSAI1RDYF**: PLLSAI1 ready interrupt flag
Set by hardware when the PLLSAI1 locks and PLLSAI1RDYDIE is set.
Cleared by software setting the PLLSAI1RDYC bit.
0: No clock ready interrupt caused by PLLSAI1 lock
1: Clock ready interrupt caused by PLLSAI1 lock

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 *RCC 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 **MSIRDYF**: MSI ready interrupt flag
Set by hardware when the MSI clock becomes stable and MSIRDYDIE is set.
Cleared by software setting the MSIRDYC bit.
0: No clock ready interrupt caused by the MSI oscillator
1: Clock ready interrupt caused by the MSI oscillator

Bit 1 **LSERDYF**: LSE ready interrupt flag
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

9.8.9 RCC clock interrupt clear register (RCC_CICR)

Address offset: 0x020

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSI48RDYC	Res.	CSSC	PLLSAI2RDYC	PLLSAI1RDYC	PLLRDYC	HSERDYC	HSIRDYC	MSIRDYC	LSERDYC	LSIRDYC
					w		w	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 Reserved, must be kept at reset value.

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

Bit 7 **PLLSAI2RDYC**: PLLSAI2 ready interrupt clear
This bit is set by software to clear the PLLSAI2RDYF flag.
0: No effect
1: Clear PLLSAI2RDYF flag

Bit 6 **PLLSAI1RDYC**: PLLSAI1 ready interrupt clear
This bit is set by software to clear the PLLSAI1RDYF flag.
0: No effect
1: Clear PLLSAI1RDYF flag

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 **MSIRDYC**: MSI ready interrupt clear
This bit is set by software to clear the MSIRDYF flag.
0: No effect
1: MSIRDYF cleared

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

9.8.10 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x028

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	FLASHRST	Res.	Res.	Res.	Res.	Res.	DMAMUX1RST	DMA2RST	DMA1RST
			rw				rw						rw	rw	rw

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

Bit 16 **TSCRST**: Touch sensing controller reset
Set and cleared by software.
0: No effect
1: Reset TSC

Bits 15:13 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 7:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1RST:** DMAMUX1 reset
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

9.8.11 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x02C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1RST	OTFDEC1RST	Res.	PKARST	RNGRST	HASHRST	AESRST
									rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADCRST	Res.	Res.	Res.	Res.	Res.	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
		rw						rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **SDMMC1RST**: SDMMC1 reset
Set and cleared by software.
0: No effect
1: Reset SDMMC1

Bit 21 **OTFDEC1RST**: OTFDEC reset
Set and cleared by software.
0: No effect
1: Reset OTFDEC

Bit 20 Reserved, must be kept at reset value.

Bit 19 **PKARST**: PKA reset
Set and cleared by software.
0: No effect
1: Reset PKA

Bit 18 **RNGRST**: Random number generator reset
Set and cleared by software.
0: No effect
1: Reset RNG

Bit 17 **HASHRST:** Hash reset
Set and cleared by software.
0: No effect
1: Reset HASH

Bit 16 **AESRST**: AES hardware accelerator reset
Set and cleared by software.
0: No effect
1: Reset AES

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ADCRST**: ADC reset
Set and cleared by software.
0: No effect
1: Reset ADC interface

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRST**: IO port H reset
Set and cleared by software.
0: No effect
1: Reset IO port H

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

9.8.12 RCC AHB3 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x030

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	OSPI1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCRST
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **OSPI1RST**: OCTOSPI1 memory interface reset
Set and cleared by software.
0: No effect
1: Reset OCTOSPI1

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

Bit 0 **FMCRST**: Flexible memory controller reset
Set and cleared by software.
0: No effect
1: Reset FMC

9.8.13 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1RST	OPAMPRST	DAC1RST	PWRRST	Res.	Res.	Res.	CRSRST	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	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
SPI3RST	SPI2RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
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 **OPAMPRST**: OPAMP interface reset
Set and cleared by software.
0: No effect
1: Reset OPAMP interface

Bit 29 **DAC1RST**: DAC1 interface reset
Set and cleared by software.
0: No effect
1: Reset DAC1 interface

Bit 28 **PWRRST**: Power interface reset
Set and cleared by software.
0: No effect
1: Reset PWR

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

Bit 24 **CRSRST**: CRS reset
Set and cleared by software.
0: No effect
1: Reset the CRS

Bit 23 **I2C3RST**: I2C3 reset
Set and reset by software.
0: No effect
1: Reset I2C3

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 13:6 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 TIM6

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

9.8.14 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x03C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RST	Res.	USBFSRST	Res.	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.	FDCAN1RST	Res.	Res.	LPTIM3RST	LPTIM2RST	Res.	Res.	Res.	I2C4RST	LPUART1RST
						rw			rw	rw				rw	rw

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

Bit 23 **UCPD1RST**: UCPD1 reset
Set and cleared by software.
0: No effect
1: Reset UCPD1

Bit 22 Reserved, must be kept at reset value.

Bit 21 **USBFSRST**: USB FS reset
Set and cleared by software.
0: No effect
1: Reset USB FS

Bits 20:10 Reserved, must be kept at reset value.

Bit 9 **FDCAN1RST**: FDCAN1 reset
Set and cleared by software.
0: No effect
1: Reset FDCAN1

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **LPTIM3RST**: LPTIM3 reset
Set and cleared by software.
0: No effect
1: Reset LPTIM3

Bit 5 **LPTIM2RST**: LPTIM2 reset
Set and cleared by software.
0: No effect
1: Reset LPTIM2

Bits 4:2 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

9.8.15 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x040

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSDM1RST	Res.	SAI2RST	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	TIM15RST
							rw		rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGRST
	rw	rw	rw	rw											rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DFSDM1RST**: Digital filters for sigma-delta modulators (DFSDM1) reset
Set and cleared by software.
0: No effect
1: Reset DFSDM1

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SAI2RST**: Serial audio interface 2 (SAI2) reset
Set and cleared by software.
0: No effect
1: Reset SAI2

Bit 21 **SAI1RST**: Serial audio interface 1 (SAI1) reset
Set and cleared by software.
0: No effect
1: Reset SAI1

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

Bit 18 **TIM17RST**: TIM17 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM17

Bit 17 **TIM16RST**: TIM16 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM16

Bit 16 **TIM15RST**: TIM15 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM15

Bit 15 Reserved, must be kept at reset value.

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

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

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRST**: SYSCFG + COMP + VREFBUF reset
0: No effect
1: Reset SYSCFG + COMP + VREFBUF

9.8.16 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x048

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.	GTZCEN	Res.	Res.	Res.	Res.	Res.	TSCEN
									rw						rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	DMAMUX1EN	DMA2EN	DMA1EN
			rw				rw						rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 22 **GTZCEN**: GTZC clock enable
Set and reset by software.
0: GTZC clock disabled
1: GTZC clock enabled

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **TSCEN**: Touch sensing controller clock enable
Set and cleared by software.
0: TSC clock disabled
1: TSC clock enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC clock enable
Set and cleared by software.
0: CRC clock disabled
1: CRC clock enabled

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 disabled
1: Flash memory interface clock enabled

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

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 disabled
1: DMA2 clock enabled

Bit 0 **DMA1EN**: DMA1 clock enable
Set and cleared by software.
0: DMA1 clock disabled
1: DMA1 clock enabled

9.8.17 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x04C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1EN	OTFDEC1EN	Res.	PKAEN	RNGEN	HASHEN	AESEN
									rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADCEN	Res.	Res.	Res.	Res.	Res.	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
		rw						rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **SDMMC1EN**: SDMMC1 clock enable
Set and cleared by software.
0: SDMMC1 clock disabled
1: SDMMC1 clock enabled

Bit 21 **OTFDEC1EN**: OTFDEC1 clock enable
Set and cleared by software.
0: OTFDEC1 clock disabled
1: OTFDEC1 clock enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **PKAEN:** PKA clock enable
Set and cleared by software.
0: PKA clock disabled
1: PKA clock enabled

Bit 18 **RNGEN:** Random Number Generator clock enable
Set and cleared by software.
0: Random Number Generator clock disabled
1: Random Number Generator clock enabled

Bit 17 **HASHEN**: HASH clock enable
Set and cleared by software
0: HASH clock disabled
1: HASH clock enabled

Bit 16 **AESEN:** AES accelerator clock enable
Set and cleared by software.
0: AES clock disabled
1: AES clock enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ADCEN:** ADC clock enable
Set and cleared by software.
0: ADC clock disabled
1: ADC clock enabled

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHEN:** IO port H clock enable
Set and cleared by software.
0: IO port H clock disabled
1: IO port H clock enabled

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

9.8.18 RCC AHB3 peripheral clock enable register(RCC_AHB3ENR)

Address offset: 0x050

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.	OSPI1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCEN
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **OSPI1EN**: OCTOSPI1 memory interface clock enable
Set and cleared by software.
0: OCTOSPI1 clock disabled
1: OCTOSPI1 clock enabled

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

Bit 0 **FMCEN**: Flexible memory controller clock enable
Set and cleared by software.
0: FMC clock disabled
1: FMC clock enabled

9.8.19 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address: 0x058

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1EN	OPAMPEN	DAC1EN	PWREN	Res.	Res.	Res.	CRSEN	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	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
SPI3EN	SPI2EN	Res.	Res.	WWDGEN	RTCAPBEN	Res.	Res.	Res.	Res.	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rw	rw			rs	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 **OPAMPEN**: OPAMP interface clock enable
Set and cleared by software.
0: OPAMP interface clock disabled
1: OPAMP interface clock enabled

Bit 29 **DAC1EN**: DAC1 interface clock enable
Set and cleared by software.
0: DAC1 interface clock disabled
1: DAC1 interface clock enabled

Bit 28 **PWREN**: Power interface clock enable
Set and cleared by software.
0: Power interface clock disabled
1: Power interface clock enabled

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

Bit 24 **CRSEN**: CRS clock enable
Set and cleared by software.
0: CRS clock disabled
1: CRS clock enabled

Bit 23 **I2C3EN**: I2C3 clock enable
Set and cleared by software.
0: I2C3 clock disabled
1: I2C3 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

Bits 9:6 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

9.8.20 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x05C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1EN	Res.	USBFSEN	Res.	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.	FDCAN1EN	Res.	Res.	LPTIM3EN	LPTIM2EN	Res.	Res.	Res.	I2C4EN	LPUART1EN
						rw			rw	rw				rw	rw

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

Bit 23 **UCPD1EN**: UCPD1 clock enable
Set and cleared by software.
0: UCPD1 clock disabled
1: UCPD1 clock enabled

Bit 22 Reserved, must be kept at reset value.

Bit 21 **USBFSEN**: USB FS clock enable
Set and cleared by software.
0: USB FS clock disabled
1: USB FS clock enabled

Bits 20:10 Reserved, must be kept at reset value.

Bit 9 **FDCAN1EN**: FDCAN1 clock enable
Set and cleared by software.
0: FDCAN1 clock disabled
1: FDCAN1 clock enabled

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **LPTIM3EN**: Low-power timer 3clock enable
Set and cleared by software.
0: LPTIM3 clock disabled
1: LPTIM3 clock enabled

Bit 5 **LPTIM2EN:** Low-power timer 2 clock enable
Set and cleared by software.
0: LPTIM2 clock disabled
1: LPTIM2 clock enabled

Bits 4:2 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 disabled
1: LPUART1 clock enabled

9.8.21 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x060

Reset value: 0x0000 0000

Access: word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSDM1EN	Res.	SAI2EN	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	TIM15EN
							rw		rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	rw	rw	rw	rw											rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DFSDM1EN**: DFSDM1 timer clock enable
Set and cleared by software.
0: DFSDM1 clock disabled
1: DFSDM1 clock enabled

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SAI2EN**: SAI2 clock enable
Set and cleared by software.
0: SAI2 clock disabled
1: SAI2 clock enabled

Bit 21 **SAI1EN**: SAI1 clock enable
Set and cleared by software.
0: SAI1 clock disabled
1: SAI1 clock enabled

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

Bit 18 **TIM17EN**: TIM17 timer clock enable
Set and cleared by software.
0: TIM17 clock disabled
1: TIM17 clock enabled

Bit 17 **TIM16EN**: TIM16 timer clock enable
Set and cleared by software.
0: TIM16 clock disabled
1: TIM16 clock enabled

Bit 16 **TIM15EN**: TIM15 timer clock enable
Set and cleared by software.
0: TIM15 clock disabled
1: TIM15 clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: USART1clock enable
Set and cleared by software.
0: USART1 clock disabled
1: USART1 clock enabled

Bit 13 T**IM8EN**: 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 clock enable
Set and cleared by software.
0: SYSCFG + COMP + VREFBUF clock disabled
1: SYSCFG + COMP + VREFBUF clock enabled

9.8.22 RCC AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB1SMENR)

Address offset: 0x068

Reset value: 0x00C1 1307

Access: no wait state, word, half-word and byte access

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.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCSMEN	Res.	Res.	SRAM1SMEN	FLASHSMEN	Res.	Res.	Res.	Res.	Res.	DMAMUX1SMEN	DMA2SMEN	DMA1SMEN
			rw			rw	rw						rw	rw	rw

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

Bit 23 **ICACHESMEN**: Instruction cache ICACHE clocks enable during Sleep and Stop modes
Set and cleared by software.
0: ICACHE clocks disabled by the clock gating during Sleep and Stop modes
1: ICACHE clocks enabled by the clock gating during Sleep and Stop modes

Bit 22 **GTZCSMEN**: GTZC clocks enable during Sleep and Stop modes
Set and cleared by software
0: GTZC clocks disabled by the clock gating during Sleep and Stop modes
1: GTZC clocks enabled by the clock gating during Sleep and Stop modes

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **TSCSMEN**: Touch sensing controller clocks enable during Sleep and Stop modes
Set and cleared by software.
0: TSC clocks disabled by the clock gating during Sleep and Stop modes
1: TSC clocks enabled by the clock gating during Sleep and Stop modes

Bits 15:13 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 during Sleep and Stop modes
1: CRC clocks enabled by the clock gating 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 during Sleep and Stop modes
1: SRAM1 interface clocks enabled by the clock gating 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 during Sleep and Stop modes
1: Flash memory interface clocks enabled by the clock gating during Sleep and Stop modes

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

Bit 2 **DMAMUX1SMEN**: DMAMUX1 clocks enable during Sleep and Stop modes.
Set and cleared by software.
0: DMAMUX1 clocks disabled by the clock gating during Sleep and Stop modes
1: DMAMUX1 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: DMA2 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: DMA1 clocks enabled by the clock gating during Sleep and Stop modes

9.8.23 RCC AHB2 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB2SMENR)

Address offset: 0x06C

Reset value: 0x006F 22FF

Access: no wait state, word, half-word and byte access

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1SMEN	OTFDEC1SMEN	Res.	PKASMEN	RNGSMEN	HASHSMEN	AESSMEN
									rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADCSMEN	Res.	Res.	Res.	SRAM2SMEN	Res.	GPIOHSMEN	GPIOGSMEN	GPIOFSMEN	GPIOESMEN	GPIODSMEN	GPIOCSMEN	GPIOBSMEN	GPIOASMEN
		rw				rw		rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **SDMMC1SMEN**: SDMMC1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SDMMC1 clocks disabled by the clock gating during Sleep and Stop modes
1: SDMMC1 clocks enabled by the clock gating during Sleep and Stop modes

Bit 21 **OTFDEC1SMEN**: OTFDEC1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: OTFDEC1 clocks disabled by the clock gating during Sleep and Stop modes
1: OTFDEC1 clocks enabled by the clock gating during Sleep and Stop modes

Bit 20 Reserved, must be kept at reset value.

Bit 19 **PKASMEN**: PKA clocks enable during Sleep and Stop modes
Set and cleared by software.
0: PKA clocks disabled by the clock gating during Sleep and Stop modes
1: PKA clocks enabled by the clock gating during Sleep and Stop modes

Bit 18 **RNGSMEN:** Random number generator (RNG) clocks enable during Sleep and Stop modes
Set and cleared by software.
0: RNG clocks disabled by the clock gating during Sleep and Stop modes
1: RNG clocks enabled by the clock gating during Sleep and Stop modes

Bit 17 **HASHSMEN**: HASH clock enable during Sleep and Stop modes
Set and cleared by software
0: HASH clocks disabled by the clock gating during Sleep and Stop modes
1: HASH clocks enabled by the clock gating during Sleep and Stop modes

Bit 16 **AESSMEN:** AES accelerator clocks enable during Sleep and Stop modes
Set and cleared by software.
0: AES clocks disabled by the clock gating during Sleep and Stop modes
1: AES clocks enabled by the clock gating during Sleep and Stop modes

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ADCSMEN:** ADC clocks enable during Sleep and Stop modes
Set and cleared by software.
0: ADC clocks disabled by the clock gating during Sleep and Stop modes
1: ADC clocks enabled by the clock gating during Sleep and Stop modes

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

Bit 9 **SRAM2SMEN:** SRAM2 interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SRAM2 interface clocks disabled by the clock gating during Sleep and Stop modes
1: SRAM2 interface clocks enabled by the clock gating during Sleep and Stop modes

Bit 8 Reserved, must be kept at reset value.

Bit 7 **GPIOHSMEN:** IO port H clocks enable during Sleep and Stop modes
Set and cleared by software.
0: IO port H clocks disabled by the clock gating during Sleep and Stop modes
1: IO port H clocks enabled by the clock gating during Sleep and Stop modes

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 during Sleep and Stop modes
1: IO port G clocks enabled by the clock gating 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 during Sleep and Stop modes
1: IO port F clocks enabled by the clock gating 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 during Sleep and Stop modes
1: IO port E clocks enabled by the clock gating 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 during Sleep and Stop modes
1: IO port D clocks enabled by the clock gating 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 during Sleep and Stop modes
1: IO port C clocks enabled by the clock gating 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 during Sleep and Stop modes
1: IO port B clocks enabled by the clock gating 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 during Sleep and Stop modes
1: IO port A clocks enabled by the clock gating during Sleep and Stop modes

9.8.24 RCC AHB3 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR)

Address offset: 0x070

Reset value: 0x0000 0101

Access: no wait state, word, half-word and byte access

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	OSPI1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCSMEN
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **OSPI1SMEN**: OCTOSPI1 memory interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: OCTOSPI1 clocks disabled by the clock gating during Sleep and Stop modes
1: OCTOSPI1 clocks enabled by the clock gating during Sleep and Stop modes

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

Bit 0 **FMCSMEN**: Flexible memory controller clocks enable during Sleep and Stop modes
Set and cleared by software.
0: FMC clocks disabled by the clock gating during Sleep and Stop modes
1: FMC clocks enabled by the clock gating during Sleep and Stop modes

9.8.25 RCC APB1 peripheral clocks enable in Sleep and Stop modes register 1 (RCC_APB1SMENR1)

Address: 0x078

Reset value: 0xF1FE CC3F

Access: no wait state, word, half-word and byte access

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1SMEN	OPAMPSMEN	DAC1SMEN	PWRSMEN	Res.	Res.	Res.	CRSSMEN	I2C3SMEN	I2C2SMEN	I2C1SMEN	UART5SMEN	UART4SMEN	USART3SMEN	USART2SMEN	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
SPI3SMEN	SPI2SMEN	Res.	Res.	WWDGSMEN	RTCAPBSMEN	Res.	Res.	Res.	Res.	TIM7SMEN	TIM6SMEN	TIM5SMEN	TIM4SMEN	TIM3SMEN	TIM2SMEN
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 during Sleep and Stop modes
1: LPTIM1 clocks enabled by the clock gating during Sleep and Stop modes

Bit 30 **OPAMPSMEN**: OPAMP interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: OPAMP interface clocks disabled by the clock gating during Sleep and Stop modes
1: OPAMP interface clocks enabled by the clock gating during Sleep and Stop modes

Bit 29 **DAC1SMEN**: DAC1 interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: DAC1 interface clocks disabled by the clock gating during Sleep and Stop modes
1: DAC1 interface clocks enabled by the clock gating during Sleep and Stop modes

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 during Sleep and Stop modes
1: Power interface clocks enabled by the clock gating during Sleep and Stop modes

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

Bit 24 **CRSSMEN**: CRS clock enable during Sleep and Stop modes
Set and cleared by software.
0: CRS clocks disabled by the clock gating during Sleep and Stop modes
1: CRS clocks enabled by the clock gating during Sleep and Stop modes

Bit 23 **I2C3SMEN**: I2C3 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: I2C3 clocks disabled by the clock gating during Sleep and Stop modes
1: I2C3 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: I2C2 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: I2C1 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: UART5 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: UART4 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: USART3 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: USART2 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: SPI3 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: SPI2 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: Window watchdog clocks enabled by the clock gating 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 during Sleep and Stop modes
1: RTC APB clock enabled by the clock gating during Sleep and Stop modes

Bits 9:6 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 during Sleep and Stop modes
1: TIM7 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: TIM6 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: TIM5 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: TIM4 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: TIM3 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: TIM2 clocks enabled by the clock gating during Sleep and Stop modes

9.8.26 RCC APB1 peripheral clocks enable in Sleep and Stop modes register 2 (RCC_APB1SMENR2)

Address offset: 0x07C

Reset value: 0x00A0 0223

Access: no wait state, word, half-word and byte access

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1SMEN	Res.	USBFSSMEN	Res.	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.	FDCAN1SMEN	Res.	Res.	LPTIM3SMEN	LPTIM2SMEN	Res.	Res.	Res.	I2C4SMEN	LPUART1SMEN
						rw			rw	rw				rw	rw

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

Bit 23 **UCPD1SMEN**: UCPD1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: UCPD1 clocks disabled by the clock gating during Sleep and Stop modes
1: UCPD1 clocks enabled by the clock gating during Sleep and Stop modes

Bit 22 Reserved, must be kept at reset value.

Bit 21 **USBFSSMEN**: USB FS clocks enable during Sleep and Stop modes
Set and cleared by software.
0: USB FS clocks disabled by the clock gating during Sleep and Stop modes
1: USB FS clocks enabled by the clock gating during Sleep and Stop modes

Bits 20:10 Reserved, must be kept at reset value.

Bit 9 **FDCAN1SMEN**: FDCAN1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: FDCAN1 clocks disabled by the clock gating during Sleep and Stop modes
1: FDCAN1 clocks enabled by the clock gating during Sleep and Stop modes

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **LPTIM3SMEN**:Low-power timer 3clocks enable during Sleep and Stop modes
Set and cleared by software.
0: LPTIM3 clocks disabled by the clock gating during Sleep and Stop modes
1: LPTIM3 clocks enabled by the clock gating during Sleep and Stop modes

Bit 5 **LPTIM2SMEN:** Low-power timer 2 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: LPTIM2 clocks disabled by the clock gating during Sleep and Stop modes
1: LPTIM2 clocks enabled by the clock gating during Sleep and Stop modes

Bits 4:2 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 during Sleep and Stop modes
1: I2C4 clocks enabled by the clock gating 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 during Sleep and Stop modes
1: LPUART1 clocks enabled by the clock gating during Sleep and Stop modes

9.8.27 RCC APB2 peripheral clocks enable in Sleep and Stop modes register (RCC_APB2SMENR)

Address: 0x080

Reset value: 0x0167 7801

Access: word, half-word and byte access

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSDM1SMEN	Res.	SAI2SMEN	SAI1SMEN	Res.	Res.	TIM17SMEN	TIM16SMEN	TIM15SMEN
							rw		rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1SMEN	TIM8SMEN	SPI1SMEN	TIM1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGSMEN
	rw	rw	rw	rw											rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DFSDM1SMEN**: DFSDM1 timer clocks enable during Sleep and Stop modes
Set and cleared by software.
0: DFSDM1 clocks disabled by the clock gating during Sleep and Stop modes
1: DFSDM1 clocks enabled by the clock gating during Sleep and Stop modes

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SAI2SMEN**: SAI2 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SAI2 clocks disabled by the clock gating during Sleep and Stop modes
1: SAI2 clocks enabled by the clock gating during Sleep and Stop modes

Bit 21 **SAI1SMEN**: SAI1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SAI1 clocks disabled by the clock gating during Sleep and Stop modes
1: SAI1 clocks enabled by the clock gating during Sleep and Stop modes

Bits 20:19 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 clocks disabled by the clock gating during Sleep and Stop modes
1: TIM17 clocks enabled by the clock gating during Sleep and Stop modes

Bit 17 **TIM16SMEN**: TIM16 timer clocks enable during Sleep and Stop modes
Set and cleared by software.
0: TIM16 clocks disabled by the clock gating during Sleep and Stop modes
1: TIM16 clocks enabled by the clock gating during Sleep and Stop modes

Bit 16 **TIM15SMEN**: TIM15 timer clocks enable during Sleep and Stop modes
Set and cleared by software.
0: TIM15 clocks disabled by the clock gating during Sleep and Stop modes
1: TIM15 clocks enabled by the clock gating during Sleep and Stop modes

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN**: USART1clocks enable during Sleep and Stop modes
Set and cleared by software.
0: USART1clocks disabled by the clock gating during Sleep and Stop modes
1: USART1clocks enabled by the clock gating 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 clocks disabled by the clock gating during Sleep and Stop modes
1: TIM8 clocks enabled by the clock gating 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 Sleep and Stop modes
1: SPI1 clocks enabled by the clock gating during Sleep and Stop modes

Bit 11 **TIM1SMEN**: TIM1 timer clocks enable during Sleep and Stop modes
Set and cleared by software.
0: TIM1 clocks disabled by the clock gating during Sleep and Stop modes
1: TIM1 clocks enabled by the clock gating during Sleep and Stop modes

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGSMEN**: SYSCFG + COMP + VREFBUF clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SYSCFG + COMP + VREFBUF clocks disabled by the clock gating during Sleep and Stop modes
1: SYSCFG + COMP + VREFBUF clocks enabled by the clock gating during Sleep and Stop modes

9.8.28 RCC peripherals independent clock configuration register 1 (RCC_CCIPR1)

Address: 0x088

Reset value: 0x0000 0000

Access: no wait states, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ADCSEL[1:0]		CLK48MSEL[1:0]		FDCANSEL[1:0]		LPTIM3SEL[1:0]		LPTIM2SEL[1:0]		LPTIM1SEL[1:0]		I2C3SEL[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
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 Reserved, must be kept at reset value.

Bits 29:28 **ADCSEL[1:0]**: ADCs 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: PLLSAI1 “R” clock (PLLADC1CLK) selected

10: reserved

11: System clock selected

Bits 27:26 **CLK48MSEL[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 FS, RNG and SDMMC.

00: HSI48 clock selected

01: PLLSAI1 “Q” clock (PLL48M2CLK) selected

10: PLL “Q” clock (PLL48M1CLK) selected

11: MSI clock selected

Bits 25:24 **FDCANSEL[1:0]**: FDCAN clock source selection

These bits are set and cleared by software to select the FDCAN kernel clock source.

00: HSE clock selected

01: PLL “Q” clock (PLL48M1CLK) selected

10: PLLSAI1 “P” clock (PLLSAI1CLK) selected

11: reserved

Bits 23:22 **LPTIM3SEL[1:0]**: Low-power timer 3 clock source selection
These bits are set and cleared by software to select the LPTIM3 clock source.
00: PCLK1 clock selected
01: LSI clock selected
10: HSI16 selected
11: LSE selected

Bits 21:20 **LPTIM2SEL[1:0]**: Low-power timer 2 clock source selection
These bits are set and cleared by software to select the LPTIM2 clock source.
00: PCLK1 selected
01: LSI clock selected
10: HSI16 clock selected
11: LSE clock selected

Bits 19:18 **LPTIM1SEL[1:0]**: Low-power timer 1 clock source selection
These bits are set and cleared by software to select the LPTIM1 clock source.
00: PCLK1 selected
01: LSI clock selected
10: HSI16 clock selected
11: LSE clock selected

Bits 17:16 **I2C3SEL[1:0]**: I2C3 clock source selection
These bits are set and cleared by software to select the I2C3 clock source.
00: PCLK1 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: reserved

Bits 15:14 **I2C2SEL[1:0]**: I2C2 clock source selection
These bits are set and cleared by software to select the I2C2 clock source.
00: PCLK1 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
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: PCLK1 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: reserved

Bits 11:10 **LPUART1SEL[1:0]**: Low-power UART1 clock source selection
These bits are set and cleared by software to select the LPUART1 clock source.
00: PCLK1 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: LSE clock selected

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: PCLK1 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: LSE clock selected

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: PCLK1 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: LSE clock selected

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: PCLK1 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: LSE clock selected

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: PCLK1 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: LSE clock selected

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: PCLK2 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: LSE clock selected

9.8.29 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x0090

Reset value: 0x0000 0000

Reset by Backup domain reset except LSCOSEL, LSCOEN and BDRST that are reset only by Backup 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 RCC Backup 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 Power control register 1 (PWR_CR1) must be set before these can be modified. Refer to Section 8.1.4: Battery backup domain for further information. These bits (except LSCOSEL,

LSCOEN and BDRST) are only reset after a Backup domain reset (see Section 9.1.3: Backup domain reset). Any internal or external reset does 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.	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
						rw	rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	LSESYSRDY	Res.	RTCSEL[1:0]		LSESYSEN	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw				r		rw	rw	rw	r	rw	rw	rw	rw	r	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **LSCOSEL**: Low-speed clock output selection
Set and cleared by software.
0: LSI clock selected
1: LSE clock selected

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable
Set and cleared by software.
0: LSCO disabled
1: LSCO enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **BDRST**: Backup domain software reset
Set and cleared by software.
0: Reset not activated
1: Reset the entire Backup domain and SRAM2

Bit 15 **RTCEN**: RTC clock enable
Set and cleared by software.
0: RTC clock disabled
1: RTC clock enabled

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

Bit 11 **LSESYSRDY**: LSE system clock (LSESYS) ready
Set and cleared by hardware to indicate when the LSE system clock is stable.When the LSESYSEN bit is set, the LSESYSRDY flag is set after two LSE clock cycles.
The LSE clock must be already enabled and stable (LSEON and LSERDY are set).
When the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.
0: LSESYS clock not ready
1: LSESYS clock ready

Bit 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 Backup 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 selected

01: LSE oscillator clock selected

10: LSI oscillator clock selected

11: HSE oscillator clock divided by 32 selected

Bit 7 **LSESYSEN**: LSE system clock (LSESYS) enable

Set by software to enable always the LSE system clock generated by RCC. This clock can be then used by any peripheral when its source clock is the LSE or at system level in case of one of the LSCOSEL, MCO, MSI PLL mode or CSS on LSE is needed.

The LSESYS clock can be generated even if LSESYSEN= 0 if the LSE clock is requested by the CSS on LSE, by a peripheral or any other source clock using LSE.

0: LSESYS only enabled when requested by a peripheral or system function

1: LSESYS enabled always generated by RCC

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

9.8.30 RCC control/status register (RCC_CSR)

Address: 0x094

Reset value: 0x0C00 0600

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
LPWRRSTF	WWDGRSTF	IWWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	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.	MSISRANGE[3:0]				Res.	Res.	Res.	LSIPRE	Res.	Res.	LSIRDY	LSION
				rw	rw	rw	rw				rw			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 **IWWDGRSTF**: 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 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MSISRANGE[3:0]**:MSI range after Standby mode

Set by software to chose the MSI frequency at startup. This range is used after exiting Standby mode until MSIRGSEL is set. After a pad or a power-on reset, the range is always 4 MHz. MSISRANGE can be written only when MSIRGSEL = 1.

0100: Range 4 around 1 MHz

0101: Range 5 around 2 MHz

0101: Range 6 around 4 MHz (reset value)

0111: Range 7 around 8 MHz

others: reserved

Note: Changing the MSISRANGE does not change the current MSI frequency.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **LSIPRE**: LSI frequency prescaler (LSI /128) enable

Set and reset by software.

This bit is used to enable the internal clock prescaler (/128) of the LSI clock. The LSIPRE bit value is only taken into account when LSI is disabled (LSION and LSIRDY bits are reset)

0: LSI clock is not divided (LSI)

1: LSI clock is divided by 128 (LSI/128).

Bits 3:2 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 CSS on LSE, by the IWDG or by the RTC.

In case LSIPRE bit is set, the LSIRDY bit is set after 0.5 LSI clock cycle (~2ms) and when LSION bit is reset, LSIRDY bit is reset after 1.5 LSI clock cycles (~6ms).

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

9.8.31 RCC clock recovery RC register (RCC_CRRCR)

Address: 0x098

Reset value: 0x0000 XXX0

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.	HSI48RDY	HSI48ON
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.

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

9.8.32 RCC peripherals independent clock configuration register 2 (RCC_CCIPR2)

Address: 0x09C

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.	OSPISEL[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.	SDMMCSEL	Res.	Res.	Res.	SAI2SEL[2:0]			SAI1SEL[2:0]			ADFSDMSEL[1:0]		DFSDMSEL	I2C4SEL[1:0]	
	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **OSPISEL[1:0]: OCTOSPI clock source selection**
Set and reset by software. This bit allows to select the OCTOSPI clock source
00: System clock selected
01: MSI clock selected
10: PLL48M1CLK clock selected
11: reserved

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **SDMMCSEL**: SDMMC clock selection
Set and reset by software.
This bit allows to select the SDMMC kernel clock source between PLLP clock (PLLSAI3CLK) or clock from internal multiplexer.
It is recommended to change this bit only after reset and before enabling the SDMMC.
0: 48 MHz clock selected
1: PLLSAI3CLK selected, in case higher than 48 MHz is needed (for SDR50 mode)

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:8 **SAI2SEL[2:0]**: SAI2 clock source selection
Set and reset by software.
If the selected clock is the external clock and this clock is stopped, it is not possible to switch to another clock. The user must switch to another clock before stopping the external clock.
000: PLLSAI1CLK clock selected
001: PLLSAI2CLK clock selected
010: PLLSAI3CLK clock selected
011: External clock SAI2_EXTCLK clock selected
100: HSI clock selected
Others: reserved

Bits 7:5 **SAI1SEL[2:0]**: SAI1 clock source selection
Set and reset by software.
If the selected clock is the external clock and this clock is stopped it is not possible to switch to another clock. The user must switch to another clock before stopping the external clock.
000: PLLSAI1CLK clock selected
001: PLLSAI2CLK clock selected
010: PLLSAI3CLK clock selected
011: External clock SAI1_EXTCLK selected
100: HSI clock selected
Others: reserved

Bits 4:3 **ADFSDMSEL[1:0]**: Digital filter for sigma-delta modulator audio clock source selection
Set and reset by software.
00: SAI1clock selected
01: HSI clock selected
10: MSI clock selected
11: reserved

Bit 2 **DFSDMSEL**: Digital filter for sigma-delta modulator kernel clock source selection
Set and reset by software.
0: APB2 clock (PCLK2) selected
1: System clock selected

Bits 1:0 **I2C4SEL[1:0]**: I2C4 clock source selection
These bits are set and cleared by software.
00: PCLK1 selected
01: System clock (SYSCLK) selected
10: HSI16 clock selected
11: reserved

9.8.33 OCTOSPI delay configuration register (RCC_DLYCFGR)

Address: 0x0A4

Reset value: 0x0000 0000h

Access: no wait state, word, half-word and byte access

This register allows to configure OCTOSPI's delay cell.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	OCTOSPI1_DLY			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **OCTOSPI1_DLY:** Delay sampling configuration on OCTOSPI1 to be used for internal sampling clock (called feedback clock) or for DQS data strobe.

Set and reset by software.

0000: 1 unitary delay
0001: 2 unitary delays
0010: 3 unitary delays
...
1111: 16 unitary delays

9.8.34 RCC secure configuration register (RCC_SECCFGR)

Address: 0x0B8

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

When the system is secure (TZEN = 1), this register provides write access security and can be read and written only by a secure access. A non-secure read and write access generates an illegal access event and data is RAZ/WI.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RMVFSEC	HSI48SEC	CLK48MSEC	PLLSAI2SEC	PLLSAI1SEC	PLLSEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	MSISEC	HSESEC	HSISEC
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RMVFSEC**: Remove reset flag security
Set and reset by software.
0: non secure
1: secure

Bit 11 **HSI48SEC**: HSI48 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 10 **CLK48MSEC**: 48 MHz clock source selection security
Set and reset by software.
0: non secure
1: secure

Bit 9 **PLLSAI2SEC**: PLLSAI2 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 8 **PLLSAI1SEC**: PLLSAI1 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 7 **PLLSEC**: main PLL clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 6 **PRESCSEC**: AHBx/APBx prescaler configuration bits security
Set and reset by software.
0: non secure
1: secure

Bit 5 **SYSCLKSEC**: SYSCLK clock selection, STOPWUCK bit, clock output on MCO configuration security
Set and reset by software.
0: non secure
1: secure

Bit 4 **LSESEC**: LSE clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 3 **LSISEC**: LSI clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 2 **MSISEC**: MSI clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 1 **HSESEC**: HSE clock configuration bits, status bits and HSE_CSS security
Set and reset by software.
0: non secure
1: secure

Bit 0 **HSISEC**: HSE clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

9.8.35 RCC secure status register (RCC_SECSR)

Address: 0x0BC

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

When the system is secure (TZEN = 1), this register provides security status of security configuration bits in RCC_SECCFGR register. Both privileged and unprivileged, accesses are allowed.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RMVFSECF	HSI48SECF	CLK48MSECF	PLLSAI2SECF	PLLSAI1SECF	PLLSECF	PRESCSECF	SYSCLKSECF	LSESECF	LSISECF	MSISECF	HSESECF	HSISECF
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RMVFSECF**: remove reset flag security flag
Set and reset by software.
0: non secure
1: secure

Bit 11 **HSI48SECF**: HSI48 clock configuration and status bits security flag
Set and reset by software.
0: non secure
1: secure

Bit 10 **CLK48MSECF**: 48 MHz clock source selection security flag
Set and reset by software.
0: non secure
1: secure

Bit 9 **PLLSAI2SECF**: PLLSAI2 clock configuration and status bits security flag
Set and reset by software.
0: non secure
1: secure

Bit 8 **PLLSAI1SECF**: PLLSAI1 clock configuration and status bits security flag
Set and reset by software.
0: non secure
1: secure

Bit 7 **PLLSECF**: main PLL clock configuration and status bits security flag
Set and reset by software.
0: non secure
1: secure

Bit 6 **PRESCSECF**: AHBx/APBx prescaler configuration bits security flag
Set and reset by software.
0: non secure
1: secure

Bit 5 **SYSCLKSECF**: SYSCLK clock selection, STOPWUCK bit, clock output on MCO configuration security flag
Set and reset by software.
0: non secure
1: secure

Bit 4 **LSESECF**: LSE clock configuration and status bits security flag
Set and reset by software.
0: non secure
1: secure

Bit 3 **LSISECF**: LSI clock configuration and status bits security flag
Set and reset by software.
0: non secure
1: secure

Bit 2 **MSISECF**: MSI clock configuration and status bits security flag
Set and reset by software.
0: non secure
1: secure

Bit 1 **HSESECF**: HSE clock configuration bits, status bits and HSE_CSS security flag
Set and reset by software.
0: non secure
1: secure

Bit 0 **HSISECF**: HSE clock configuration and status bits security flag
Set and reset by software.
0: non secure
1: secure

9.8.36 RCC AHB1 security status register (RCC_AHB1SECSR)

Address: 0x0E8

Reset value: 0x0040 0300

Access: no wait state, word, half-word and byte access

When the system is secure (TZEN =1), this register provides AHB1 peripheral clock security status. When a peripheral is configured as secure, its clock is also secure.

Both privileged and unprivileged, secure and non-secure accesses are allowed.

When the system is not secure (TZEN = 0), this register is RAZ.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCSECF	Res.	Res.	SRAM1SECF	FLASHSECF	Res.	Res.	Res.	Res.	Res.	DMAMUX1SECF	DMA2SECF	DMA1SECF
			r			r	r						r	r	r

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

Bit 23 **ICACHESECF**: Instruction cache (ICACHE) clock security flag
This flag is set by hardware when ICACHE is secure.
0: non secure
1: secure

Bit 22 **GTZCSECF**: GTZC controller clock security flag
This flag is set by hardware when GTZC is secure.
0: non secure
1: secure

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **TSCSECF**: Touch sensing controller (TSC) clock security flag
This flag is set by hardware when TSC is secure.
0: non secure
1: secure

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCSECF**: CRC clock security flag
This flag is set by hardware when CRC is secure.
0: non secure
1: secure

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAM1SECF**: SRAM1 clock security flag
This flag is set by hardware when SRAM1 is secure.
0: non secure
1: secure

Bit 8 **FLASHSECF**: Flash memory interface clock security flag
This flag is set by hardware when Flash memory is secure.
0: non secure
1: secure

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

Bit 2 **DMAMUX1SECF**: DMAMUX1 clock security flag
This flag is set by hardware when DMAMUX1 is secure.
0: non secure
1: secure

Bit 1 **DMA2SECF**: DMA2 clock security flag
This flag is set by hardware when DMA2 is secure.
0: non secure
1: secure

Bit 0 **DMA1SECF**: DMA1 clock security flag
This flag is set by hardware when DMA1 is secure.
0: non secure
1: secure

9.8.37 RCC AHB2 security status register (RCC_AHB2SECSR)

Address: 0x0EC

Reset value: 0x0020 02FF

Access: no wait state, word, half-word and byte access

When the system is secure (TZEN =1), this register provides AHB2 peripheral clock security status. When a peripheral is configured as secure, its clock is also secure.

Both privileged and unprivileged, secure and non-secure accesses are allowed.

When the system is not secure (TZEN = 0), this register is RAZ.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1SECF	OTFDEC1SECF	Res.	Res.	Res.	Res.	Res.
									r	r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SRAM2SECF	Res.	GPIOHSECF	GPIOGSECF	GPIOFSECF	GPIOESECF	GPIODSECF	GPIOCSECF	GPIOBSECF	GPIOASECF
						r		r	r	r	r	r	r	r	r

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

Bit 22 **SDMMC1SECF**: SDMMC1 clock security flag
This flag is set by hardware when SDMMC1 is secure.
0: non secure
1: secure

Bit 21 **OTFDEC1SECF**: OTFDEC1 controller clock security flag
This flag is set by hardware when OTFDEC1 is secure.
0: non secure
1: secure

Bits 20:10 Reserved, must be kept at reset value.

Bit 9 **SRAM2SECF**: SRAM2 clock security flag
This flag is set by hardware when SRAM2 is secure.
0: non secure
1: secure

Bit 8 Reserved, must be kept at reset value.

Bit 7 **GPIOHSECF**: GPIOH clock security flag
This flag is set by hardware when GPIOH is secure.
0: non secure
1: secure

Bit 6 **GPIOGSECF**: GPIOG clock security flag
This flag is set by hardware when GPIOHG is secure.
0: non secure
1: secure

Bit 5 **GPIOFSECF**: GPIOF clock security flag
This flag is set by hardware when GPIOHF is secure.
0: non secure
1: secure

Bit 4 **GPIOESECF**: GPIOE clock security flag
This flag is set by hardware when GPIOE is secure.
0: non secure
1: secure

Bit 3 **GPIODSECF**: GPIOD clock security flag
This flag is set by hardware when GPIOD is secure.
0: non secure
1: secure

Bit 2 **GPIOCSECF**: GPIOC clock security flag
This flag is set by hardware when GPIOC is secure.
0: non secure
1: secure

Bit 1 **GPIOBSECF**: GPIOB clock security flag
This flag is set by hardware when GPIOB is secure.
0: non secure
1: secure

Bit 0 **GPIOASECF**: GPIOA clock security flag
This flag is set by hardware when GPIOA is secure.
0: non secure
1: secure

9.8.38 RCC AHB3 security status register (RCC_AHB3SECSR)

Address: 0x0F0

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

When the system is secure (TZEN =1), this register provides AHB3 peripheral clock security status. When a peripheral is configured as secure, its clock is also secure.

Both privileged and unprivileged, secure and non-secure accesses are allowed.

When the system is not secure (TZEN = 0), this register is RAZ.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	OSPI1SECF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCSECF
							r								r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **OSPI1SECF**: OCTOSPI1 clock security flag
This flag is set by hardware when OCTOSPI1 is secure.
0: non secure
1: secure

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

Bit 0 **FMCSECF**: FMC clock security flag
This flag is set by hardware when FMC is secure.
0: non secure
1: secure

9.8.39 RCC APB1 security status register 1 (RCC_APB1SECSR1)

Address: 0x0F8

Reset value: 0x0000 0400

Access: no wait state, word, half-word and byte access

When the system is secure (TZEN = 1), this register provides APB1 peripheral clock security status. When a peripheral is configured as secure, its clock is also secure.

Both privileged and unprivileged, secure and non-secure accesses are allowed.

When the system is not secure (TZEN = 0), this register is RAZ.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1SECF	OPAMPSECF	DAC1SECF	PWRSECF	Res.	Res.	Res.	CRSSECF	I2C3SECF	I2C2SECF	I2C1SECF	UART5SECF	UART4SECF	UART3SECF	UART2SECF	Res.
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
SPI3SECF	SPI2SECF	Res.	Res.	WWDGSECF	RTCAPBSECF	Res.	Res.	Res.	Res.	TIM7SECF	TIM6SECF	TIM5SECF	TIM4SECF	TIM3SECF	TIM2SECF
r	r			r	r					r	r	r	r	r	r

Bit 31 **LPTIM1SECF**: LPTIM1 clock security flag
This flag is set by hardware when LPTIM1 is secure.
0: non secure
1: secure

Bit 30 **OPAMPSECF**: OPAMP clock security flag
This flag is set by hardware when OPAMP is secure.
0: non secure
1: secure

Bit 29 **DAC1SECF**: DAC1 clock security flag
This flag is set by hardware when DAC1 is secure.
0: non secure
1: secure

Bit 28 **PWRSECF**: PWR clock security flag
This flag is set by hardware when PWR is secure.
0: non secure
1: secure

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

Bit 24 **CRSSECF**: CRS clock security flag
This flag is set by hardware when CRS is secure.
0: non secure
1: secure

Bit 23 **I2C3SECF**: I2C3 clock security flag
This flag is set by hardware when I2C3 is secure.
0: non secure
1: secure

Bit 22 **I2C2SECF**: I2C2 clock security flag
This flag is set by hardware when I2C2 is secure.
0: non secure
1: secure

Bit 21 **I2C1SECF**: I2C1 clock security flag
This flag is set by hardware when I2C1 is secure.
0: non secure
1: secure

Bit 20 **UART5SECF**: UART5 clock security flag
This flag is set by hardware when UART5 is secure.
0: non secure
1: secure

Bit 19 **UART4SECF**: UART4 clock security flag
This flag is set by hardware when UART4 is secure.
0: non secure
1: secure

Bit 18 **UART3SECF**: UART3 clock security flag
This flag is set by hardware when UART3 is secure.
0: non secure
1: secure

Bit 17 **UART2SECF**: UART2 clock security flag
This flag is set by hardware when UART2 is secure.
0: non secure
1: secure

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3SECF**: SPI3 clock security flag
This flag is set by hardware when SPI3 is secure.
0: non secure
1: secure

Bit 14 **SPI2SECF**: SPI2 clock security flag
This flag is set by hardware when SPI2 is secure.
0: non secure
1: secure

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGSECF**: WWDG clock security flag
This flag is set by hardware when WWDG is secure.
0: non secure
1: secure

Bit 10 **RTCAPBSECF**: RTC APB clock security flag
This flag is set by hardware when RTC APB is secure.
0: non secure
1: secure

Bits 9:6 Reserved, must be kept at reset value.

Bit 5 **TIM7SECF**: TIM7 clock security flag
This flag is set by hardware when TIM7 is secure.
0: non secure
1: secure

Bit 4 **TIM6SECF**: TIM6 clock security flag
This flag is set by hardware when TIM6 is secure.
0: non secure
1: secure

Bit 3 **TIM5SECF**: TIM5 clock security flag
This flag is set by hardware when TIM5 is secure.
0: non secure
1: secure

Bit 2 **TIM4SECF**: TIM4 clock security flag
This flag is set by hardware when TIM4 is secure.
0: non secure
1: secure

Bit 1 **TIM3SECF**: TIM3 clock security flag
This flag is set by hardware when TIM3 is secure.
0: non secure
1: secure

Bit 0 **TIM2SECF**: TIM2 clock security flag
This flag is set by hardware when TIM2 is secure.
0: non secure
1: secure

9.8.40 RCC APB1 security status register 2 (RCC_APB1SECSR2)

Address: 0x0FC

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

When the system is secure (TZEN = 1), this register provides APB1 peripheral clock security status. When a peripheral is configured as secure, its clock is also secure.

Both privileged and unprivileged, secure and non-secure accesses are allowed.

When the system is not secure (TZEN = 0), this register is RAZ.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1SECF	Res.	USBFSSECF	Res.	Res.	Res.	Res.	Res.
								r		r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1SECF	Res.	Res.	LPTIM3SECF	LPTIM2SECF	Res.	Res.	Res.	I2C4SECF	LPUART1SECF
						r			r	r				r	r

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

Bit 23 **UCPD1SECF**: UCPD1 clock security flag
This flag is set by hardware when UCPD1 is secure.
0: non secure
1: secure

Bit 22 Reserved, must be kept at reset value.

Bit 21 **USBFSSECF**: USB FS clock security flag
This flag is set by hardware when USB FS is secure.
0: non secure
1: secure

Bits 20:10 Reserved, must be kept at reset value.

Bit 9 **FDCAN1SECF**: FDCAN1 clock security flag
This flag is set by hardware when FDCAN1 is secure.
0: non secure
1: secure

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **LPTIM3SECF**: LPTIM3 clock security flag
This flag is set by hardware when LPTIM3 is secure.
0: non secure
1: secure

Bit 5 **LPTIM2SECF**: LPTIM2 clock security flag
This flag is set by hardware when LPTIM2 is secure.
0: non secure
1: secure

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **I2C4SECF**: I2C4 clock security flag
This flag is set by hardware when I2C4 is secure.
0: non secure
1: secure

Bit 0 **LPUART1SECF**: LPUART1 clock security flag
This flag is set by hardware when LPUART1 is secure.
0: non secure
1: secure

9.8.41 RCC APB2 security status register (RCC_APB2SECSR)

Address: 0x100

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

When the system is secure (TZEN = 1), this register provides APB2 peripheral clock security status. When a peripheral is configured as secure, its clock is also secure.

Both privileged and unprivileged, secure and non-secure accesses are allowed.

When the system is not secure (TZEN = 0), this register is RAZ.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSDM1SECF	Res.	SAI2SECF	SAI1SECF	Res.	Res.	TIM17SECF	TIM16SECF	TIM15SECF
							r		r	r			r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1SECF	TIM8SECF	SPI1SECF	TIM1SECF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGSECF
	r	r	r	r											r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DFSDM1SECF**: DFSDM1 clock security flag
This flag is set by hardware when DFSDM1 is secure.
0: non secure
1: secure

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SAI2SECF**: SAI2 clock security flag
This flag is set by hardware when SAI2 is secure.
0: non secure
1: secure

Bit 21 **SAI1SECF**: SAI1 clock security flag
This flag is set by hardware when SAI1 is secure.
0: non secure
1: secure

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

Bit 18 **TIM17SECF**: TIM17 clock security flag
This flag is set by hardware when TIM17 is secure.
0: non secure
1: secure

Bit 17 **TIM16SECF**: TIM16 clock security flag
This flag is set by hardware when TIM16 is secure.
0: non secure
1: secure

Bit 16 **TIM15SECF**: TIM15 clock security flag
This flag is set by hardware when TIM15 is secure.
0: non secure
1: secure

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1SECF**: USART1 clock security flag
This flag is set by hardware when USART1 is secure.
0: non secure
1: secure

Bit 13 **TIM8SECF**: TIM8 clock security flag
This flag is set by hardware when TIM8 is secure.
0: non secure
1: secure

Bit 12 **SPI1SECF**: SPI1 clock security flag
This flag is set by hardware when SPI1 is secure.
0: non secure
1: secure

Bit 11 **TIM1SECF**: TIM1 clock security flag
This flag is set by hardware when TIM1 is secure.
0: non secure
1: secure

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGSECF**: SYSCFG clock security flag
This flag is set by hardware when SYSCFG is secure.
0: non secure
1: secure

9.8.42 RCC register map

The following table gives the RCC register map and the reset values.

Table 80. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	PRIV	Res.	PLLSAI2RDY	PLLSAI2ON	PLLSAI1RDY	PLLSAI1ON	PLLRDY	PLLON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON	Res.	Res.	Res.	Res.	HSIASFS	HSIRDY	HSIKERON	HSION	MSIRANGE[3:0]				MSIRGSEL	MSIPLLEN	MSIRDY	MSION
	Reset value	0		0	0	0	0	0	0					0	0	0	0					0	0	0	0	0	1	1	0	0	0	1	1
0x004	RCC_ICSCR	Res.	HSITRIM[6:0]							HSICAL[7:0]								MSITRIM[7:0]								MSICAL[7:0]							
	Reset value		1	0	0	0	0	0	0	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x
0x008	RCC_CFGR	Res.	MCOPRE[2:0]			MCOSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPWUCK	Res.	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
	Reset value		0	0	0	0	0	0	0									0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RCC_ PLLCFGR	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.	PLLSRC[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
0x010	RCC_ PLLSAI1 CFGR	PLLSAI1PDIV[4:0]					PLLSAI1R[1:0]		PLLSAI1REN	Res.	PLLSAI1Q[1:0]		PLLSAI1QEN	Res.	Res.	PLLSAI1P	PLLSAI1PEN	Res.	PLLSAI1N[6:0]							PLLSAI1M[3:0]				Res.	Res.	PLLSAI1SRC[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
0x014	RCC_ PLLSAI2 CFGR	PLLSAI2PDIV[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLSAI2P	PLLSAI2PEN	Res.	PLLSAI2N[6:0]							PLLSAI2M[3:0]				Res.	Res.	PLLSAI2SRC[1:0]	
	Reset value	0	0	0	0	0										0	0		0	0	1	0	0	0	0	0	0	0	0			0	0
0x018	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48RDYIE	Res.	Res.	PLLSAI2RDYIE	PLLSAI1RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	MSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value																						0			0	0	0	0	0	0	0	0

Table 80. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48RDYF	Res.	CSSF	PLLSAI2RDYF	PLLSAI1RDYF	PLLRDYF	HSERDYF	HSIRDYF	MSIRDYF	LSERDYF	LSIRDYF
	Reset value																						0		0	0	0	0	0	0	0	0	0
0x020	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48RDYC	Res.	CSSC	PLLSAI2RDYC	PLLSAI1RDYC	PLLRDYC	HSERDYC	HSIRDYC	MSIRDYC	LSERDYC	LSIRDYC
	Reset value																						0		0	0	0	0	0	0	0	0	0
0x028	RCC_ AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCRST	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	FLASHRST	Res.	Res.	Res.	Res.	Res.	DMAMUX1RST	DMA2RST	DMA1RST
	Reset value																0				0				0						0	0	0
0x02C	RCC_ AHB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1RST	OTFDEC1RST	Res.	PKARST	RNGRST	HASHRST	AESRST	Res.	Res.	ADCRST	Res.	Res.	Res.	Res.	GPIOHRST	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
0x030	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.	OSPI1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCRST
	Reset value																								0								0
0x034	Reserved	Reserved																															
0x038	RCC_ APB1RSTR1	LPTIM1RST	OPAMPRST	DAC1RST	PWRRST	Res.	Res.	Res.	CRSRST	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	Res.	Res.	Res	Res.	Res.	Res.	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
0x03C	RCC_ APB1RSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RST	Res.	USBFSRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	FDCAN1RST	Res.	Res.	LPTIM3RST	LPTIM2RST	Res.	Res.	Res.	I2C4RST	LPUART1RST
	Reset value									0		0												0			0	0				0	0

Table 80. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	RCC_ APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSDM1RST	Res.	SAI2RST	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	Res.	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
0x048	RCC_ AHB1ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GTZCEN	Res.	Res.	Res.	Res.	Res.	TSCEN	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	DMAMUX1EN	DMA2EN	DMA1EN
	Reset value										0						0				0				1						0	0	0
0x04C	RCC_ AHB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1EN	OTFDEC1EN	Res.	PKAEN	RNGEN	HASHEN	AESEN	Res.	Res.	ADCEN	Res.	Res.	Res.	Res.	Res.	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value										0	0		0	0	0	0			0						0	0	0	0	0	0	0	0
0x050	RCC_ AHB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPI1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCEN
	Reset value																								0								0
0x054	Reserved	Reserved																															
0x058	RCC_ APB1ENR1	LPTIM1EN	OPAMPEN	DAC1EN	PWREN	Res.	Res.	Res.	CRSEN	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	Res.	WWDGEN	RTCAPBEN	Res.	Res.	Res.	Res.	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0	0	0	0				0	0	0	0	0	0	0	0		0	0			0	0				0	0	0	0	0	0
0x05C	RCC_ APB1ENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1EN	Res.	USBFSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1EN	Res.	Res.	Res.	LPTIM3EN	LPTIM2EN	Res.	Res.	Res.	I2C4EN	LPUART1EN
	Reset value									0		0												0				0	0				0	0
0x060	RCC_ APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSDM1EN	Res.	SAI2EN	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	Reset value								0		0	0			0	0	0		0	0	0	0												0
0x064	Reserved	Reserved																															

Table 80. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x068	RCC_AHB1 SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ICACHESMEN	GTZCSMEN	Res.	Res.	Res.	Res.	Res.	TSCSMEN	Res.	Res.	Res.	CRCSMEN	Res.	Res.	SRAM1SMEN	FLASHSMEN	Res.	Res.	Res.	Res.	Res.	DMAMUX1SMEN	DMA2SMEN	DMA1SMEN
	Reset value									1	1						1				1			1	1						1	1	1
0x06C	RCC_AHB2 SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1SMEN	OTFDEC1SMEN	Res.	PKASMEN	RNGSMEN	HASHSMEN	AESSMEN	Res.	Res.	ADCSMEN	Res.	Res.	Res.	SRAM2SMEN	Res.	GPIOHSMEN	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
0x070	RCC_AHB3 SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPI1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCSMEN
	Reset value																								1								1
0x074	Reserved	Reserved																															
0x078	RCC_APB1 SMENR1	LPTIM1SMEN	OPAMPSMEN	DAC1SMEN	PWRSMEN	Res.	Res.	Res.	CRSSMEN	I2C3SMEN	I2C2SMEN	I2C1SMEN	UART5SMEN	UART4SMEN	USART3SMEN	USART2SMEN	Res.	SPI3SMEN	SPI2SMEN	Res.	Res.	WWDGSMEN	RTCAPBSMEN	Res.	Res.	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
0x07C	RCC_APB1 SMENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1SMEN	Res.	USBSFSSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1SMEN	Res.	Res.	LPTIM3SMEN	LPTIM2SMEN	Res.	Res.	Res.	I2C4SMEN	LPUART1SMEN
	Reset value									1		1												1			1	1				1	1
0x080	RCC_APB2 SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSDM1SMEN	Res.	SAI2SMEN	SAI1SMEN	Res.	Res.	TIM17SMEN	TIM16SMEN	TIM15SMEN	Res.	USART1SMEN	TIM8SMEN	SPI1SMEN	TIM1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGSMEN
	Reset value								1		1	1			1	1	1		1	1	1	1												1
0x084	Reserved	Reserved																															

Table 80. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088	RCC_ CCIPR1	Res.	Res.	ADCSEL[1:0]		CLK48MSEL[1:0]		FDCANSEL[1:0]		LPTIM3SEL[1:0]		LPTIM2SEL[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
0x090	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	Res.	LSESYSRDY	Res.	RTCSEL[1:0]		LSESYSEN	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
	Reset value							0	0								0	0				0		0	0	0	0	0	0	0	0	0
0x094	RCC_CSR	LPWRRSTF	WWDGRSTF	IWWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSISRANGE[3:0]				Res.	Res.	Res.	LSIPRE	Res.	Res.	LSIRDY	LSION
	Reset value	0	0	0	0	1	0	1		0												0	1	1	0				0			0	0
0x098	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
0x09C	RCC_ CCIPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPISEL[1:0]		Res.	Res.	Res.	Res.	Res.	SDMMCSEL	Res.	Res.	Res.	SAI2SEL[2:0]			SAI1SEL[2:0]			ADFSDMSEL[1:0]		DFSDMSEL	I2C4SEL[1:0]	
	Reset value											0	0						0				0	0	0	0	0	0	0	0	0	0	0
0x0A0	Reserved	Reserved																															
0xA4	RCC_DLY CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI 1_DLY			
	Reset value																													0	0	0	0
0x0A8 to 0x0B4	Reserved	Reserved																															
0x0B8	RCC_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVFSEC	HSI48SEC	CLK48MSEC	PLLSAI2SEC	PLLSAI1SEC	PLLSEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	MSISEC	HSESEC	HSISEC
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0

Table 80. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0BC	RCC_ SECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVFSECF	HSI48SECF	CLK48MSECF	PLLSAI2SECF	PLLSAI1SECF	PLLSECF	PRESCSECF	SYSCLKSECF	LSESECF	LSISECF	MSISECF	HSESECF	HSISECF
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C0 to 0x0E4	Reserved	Reserved																															
0x0E8	RCC_ AHB1SECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ICACHESECF	GTZCSECF	Res.	Res.	Res.	Res.	Res.	TSCSECF	Res.	Res.	Res.	CRCESECF	Res.	Res.	SRAM1SECF	FLASHSECF	Res.	Res.	Res.	Res.	Res.	DMAMUX1SECF	DMA2SECF	DMA1SECF
	Reset value									0	1						0				0			1	1						0	0	0
0x0EC	RCC_ AHB2SECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1SECF	OTFDEC1SECF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2SECF	Res.	GPIOHSECF	GPIOGSECF	GPIOFSECF	GPIOESECF	GPIODSECF	GPIOCSECF	GPIOBSECF	GPIOASECF
	Reset value										0	1												1		1	1	1	1	1	1	1	1
0x0F0	RCC_ AHB3SECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPI1SECF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCSECF
	Reset value																								0								0
0x0F4	Reserved	Reserved																															
0x0F8	RCC_APB1 SECSR1	LPTIM1SECF	OPAMPSECF	DAC1SECF	PWRSECF	Res.	Res.	Res.	CRSSECF	I2C3SECF	I2C2SECF	I2C1SECF	UART5SECF	UART4SECF	UART3SECF	UART2SECF	Res.	SPI3SECF	SPI2SECF	Res.	Res.	WWDGSECF	RTCAPBSECF	Res.	Res.	Res.	Res.	TIM7SECF	TIM6SECF	TIM5SECF	TIM4SECF	TIM3SECF	TIM2SECF
	Reset value	0	0	0	0				0	0	0	0	0	0	0	0		0	0			0	1					0	0	0	0	0	0
0x0FC	RCC_APB1 SECSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1SECF	Res.	USBFSSECF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1SECF	Res.	Res.	LPTIM3SECF	LPTIM2SECF	Res.	Res.	Res.	I2C4SECF	LPUART1SECF
	Reset value									0		0												0			0	0				0	0

Table 80. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100	RCC_ APB2SECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSDM1SECF	Res.	SAI2SECF	SAI1SECF	Res.	Res.	TIM17SECF	TIM16SECF	TIM15SECF	Res.	USART1SECF	TIM8SECF	SPI1SECF	TIM1SECF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGSECF
	Reset value								0		0	0			0	0	0		0	0	0	0											0

Refer to *Section 2.3 on page 87* for the register boundary addresses.

10 Clock recovery system (CRS)

10.1 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.

10.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - External pin
  - LSE oscillator output
  - USB SOF 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)

10.3 CRS implementation

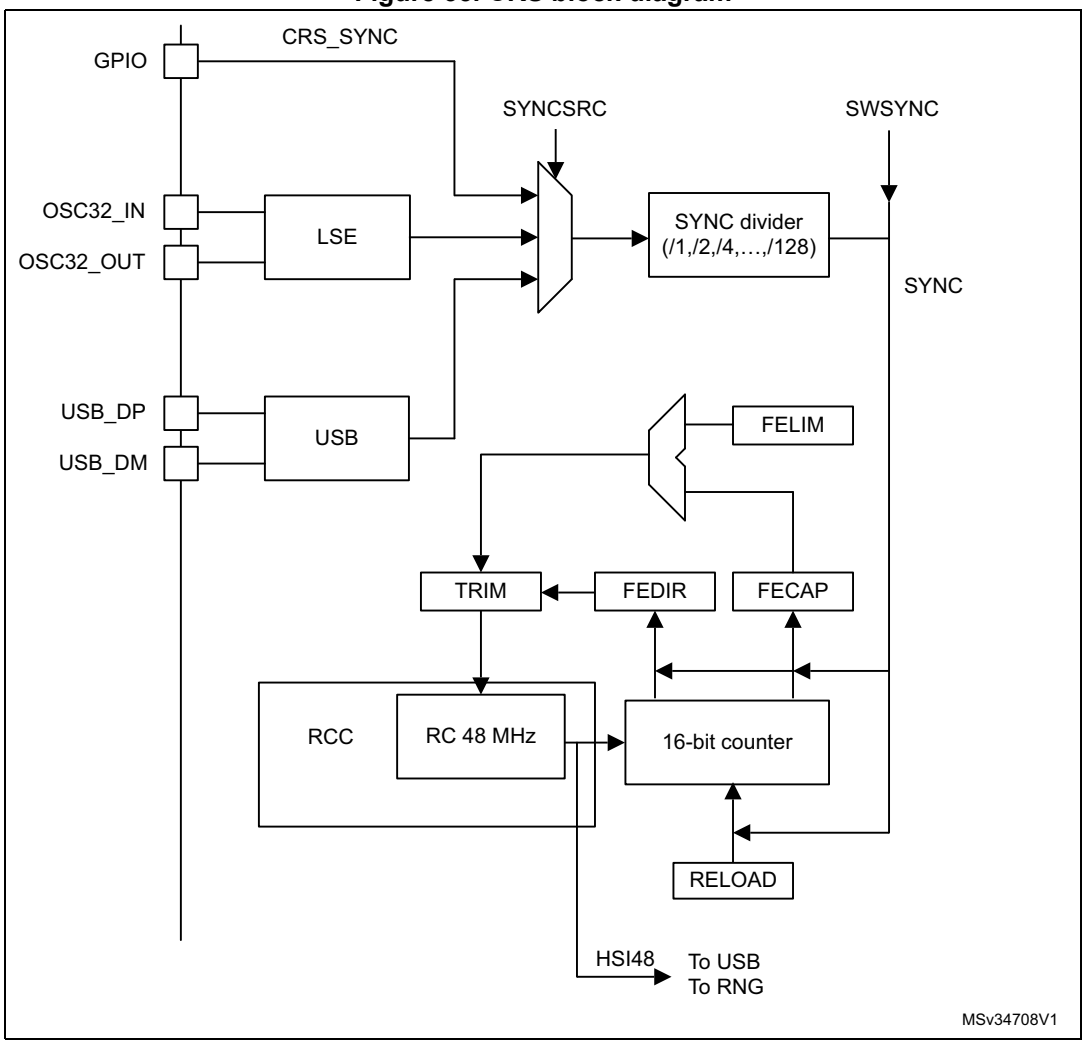
Table 81. CRS features

Feature	CRS1
TRIM width	7 bits

10.4 CRS functional description

10.4.1 CRS block diagram

Figure 35. CRS block diagram



10.4.2 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 10.7.2: CRS configuration register (CRS_CFGR)*.

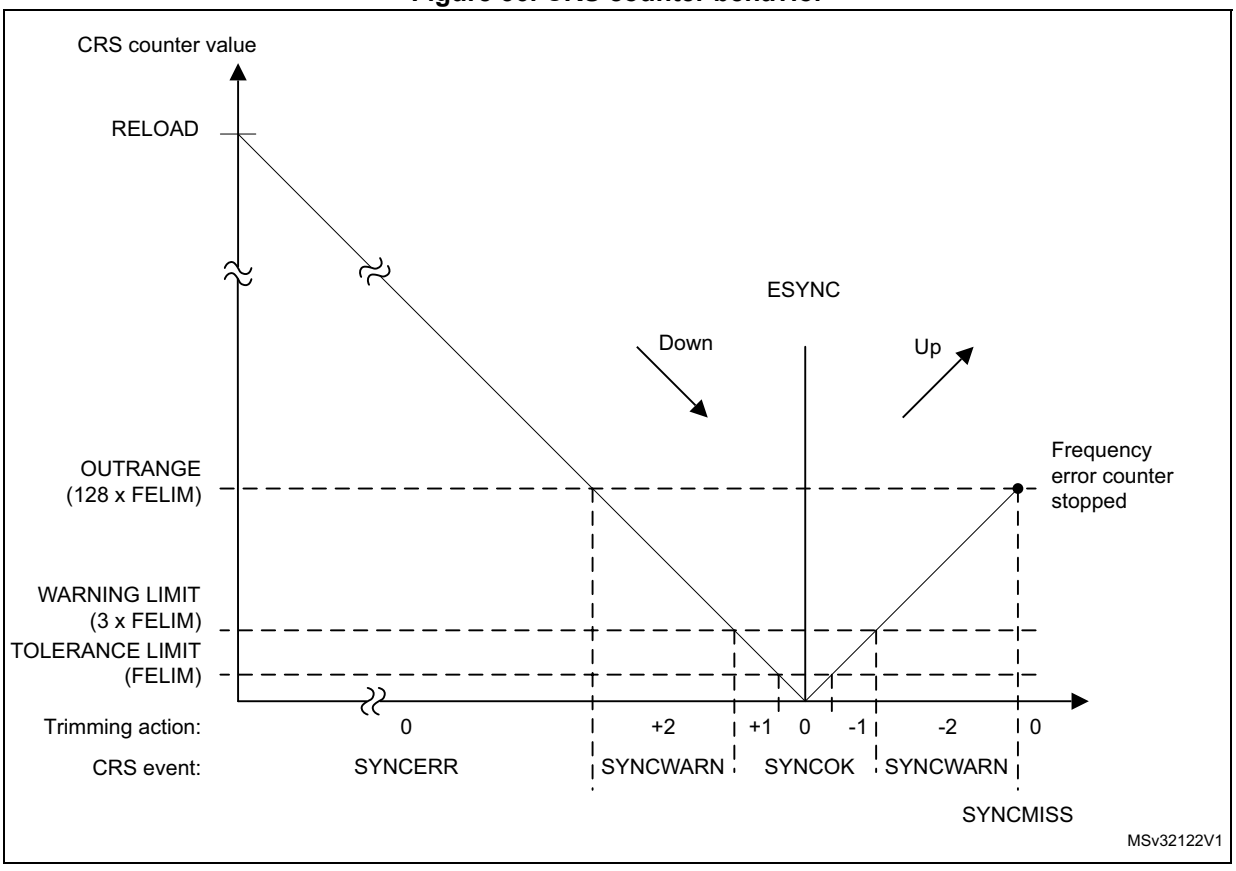
It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

10.4.3 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 36. CRS counter behavior



10.4.4 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.

10.4.5 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:

$$RELOAD = (f_{TARGET} / f_{SYNC}) - 1$$

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:

$$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$$

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).

10.5 CRS low-power modes

Table 82. 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.

10.6 CRS interrupts

Table 83. 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

10.7 CRS registers

Refer to *Section 1.2 on page 76* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

10.7.1 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 10.3*), 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 10.3*) 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 10.4.4* 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

10.7.2 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 10.4.4* 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 10.4.3* for more details about counter behavior.

10.7.3 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 10.4.4* 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

10.7.4 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.

10.7.5 CRS register map

Table 84. 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 2.3 on page 87* for the register boundary addresses.

11 General-purpose I/Os (GPIO)

11.1 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) and a secure configuration register (GPIOx_SECCFGR).

11.2 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
- TrustZone security support

11.3 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 37 and *Figure 38* show the basic structures of a standard and a 5-Volt tolerant I/O port bit, respectively. *Table 85* gives the possible port bit configurations.

Figure 37. Basic structure of an I/O port bit

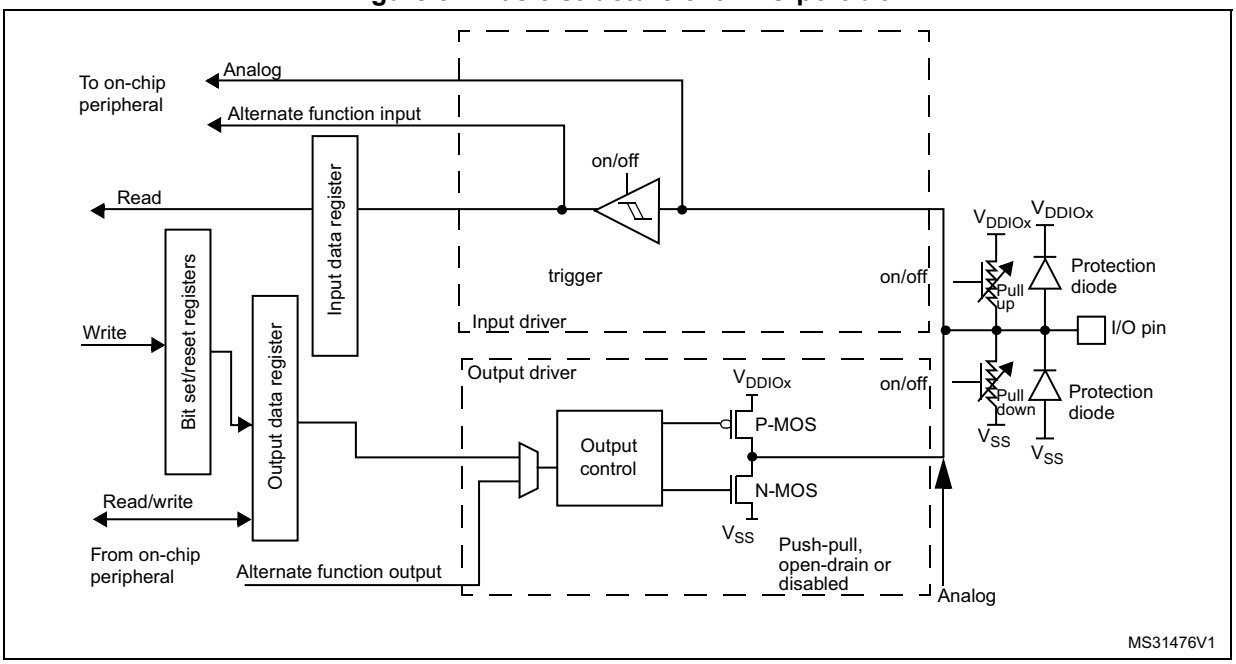
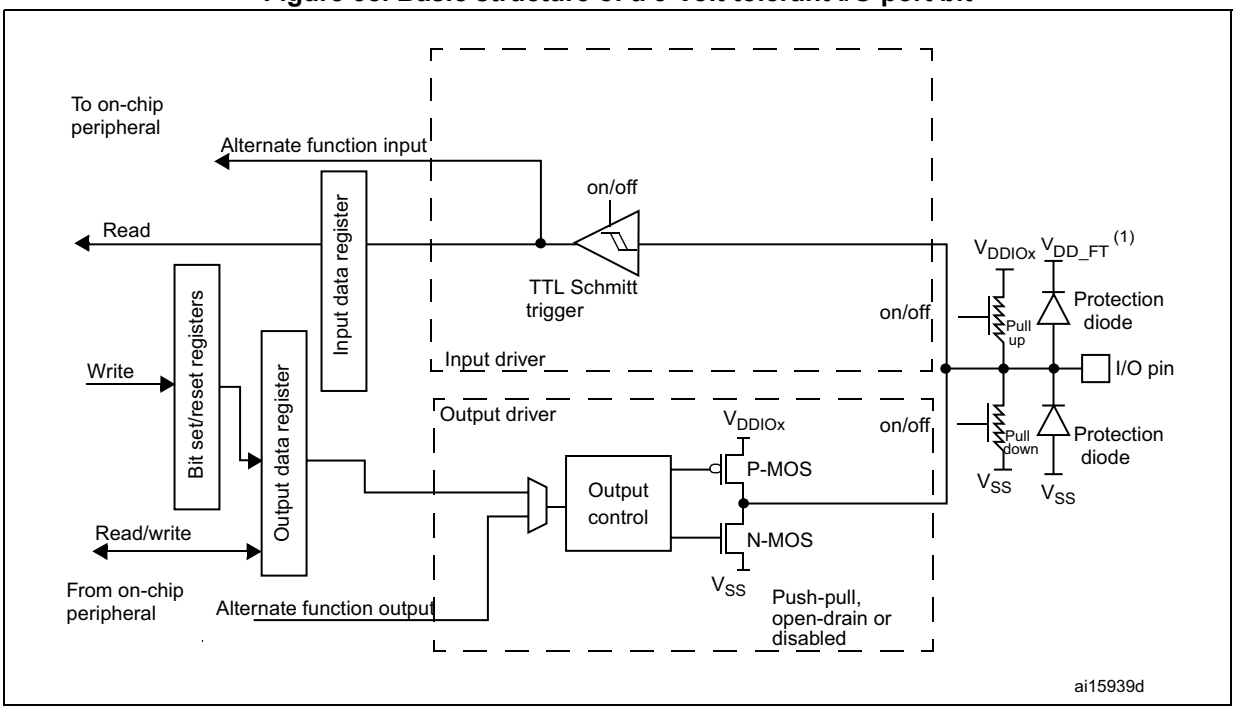


Figure 38. Basic structure of a 5-Volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Table 85. Port bit configuration table(1)

MODE(i) [1:0]	OTYPER(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		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

11.3.1 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

PH3/BOOT0 is in input mode during the reset until at least the end of the option byte loading phase. See *Section 11.3.15: Using PH3 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.

11.3.2 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 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.

11.3.3 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.

11.3.4 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 11.6.5: GPIO port input data register (GPIOx_IDR) (x = A to H)* and *Section 11.6.6: GPIO port output data register (GPIOx_ODR) (x = A to H)* for the register descriptions.

11.3.5 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.

11.3.6 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 11.6.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to H)*) 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 11.6.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to H)*.

11.3.7 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.

11.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output or alternate function mode (the port must not be configured in analog mode).

Refer to *Section 17: Extended interrupts and event controller (EXTI).*

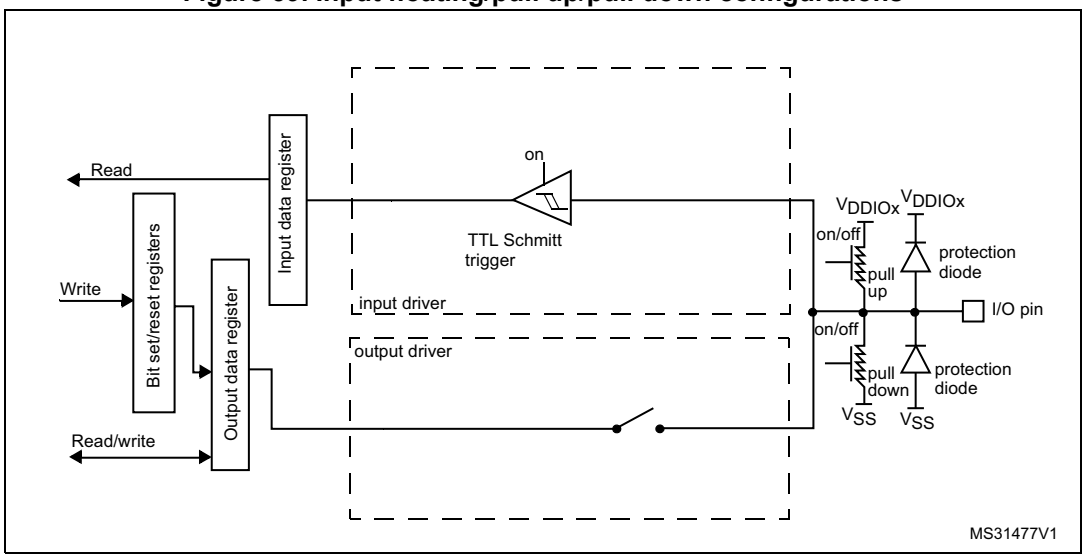
11.3.9 Input configuration

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 39 shows the input configuration of the I/O port bit.

Figure 39. Input floating/pull up/pull down configurations



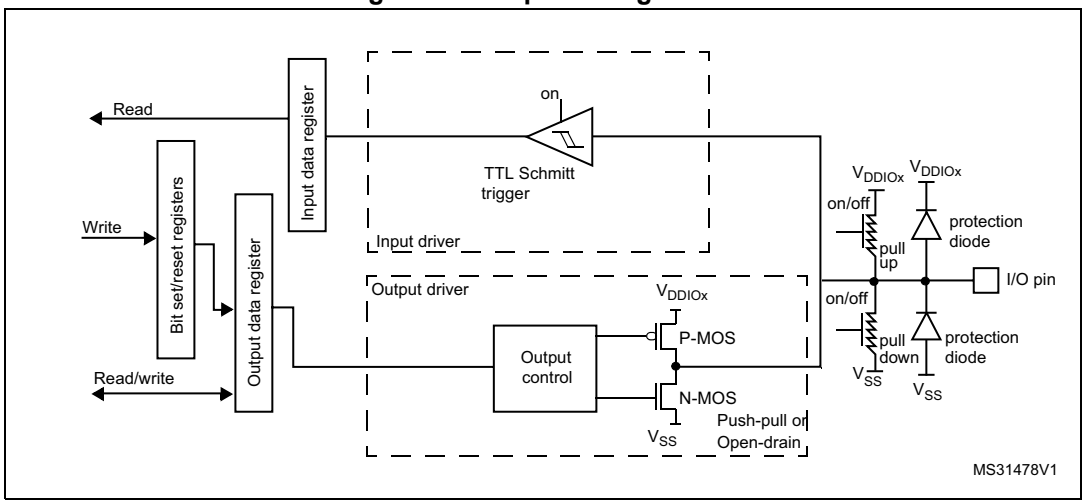
11.3.10 Output configuration

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 40 shows the output configuration of the I/O port bit.

Figure 40. Output configuration



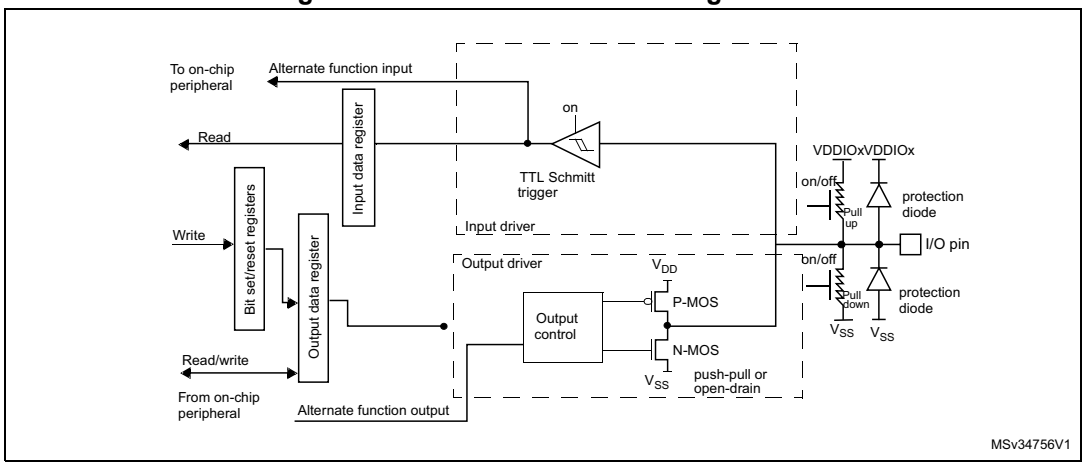
11.3.11 Alternate function configuration

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 41 shows the Alternate function configuration of the I/O port bit.

Figure 41. Alternate function configuration



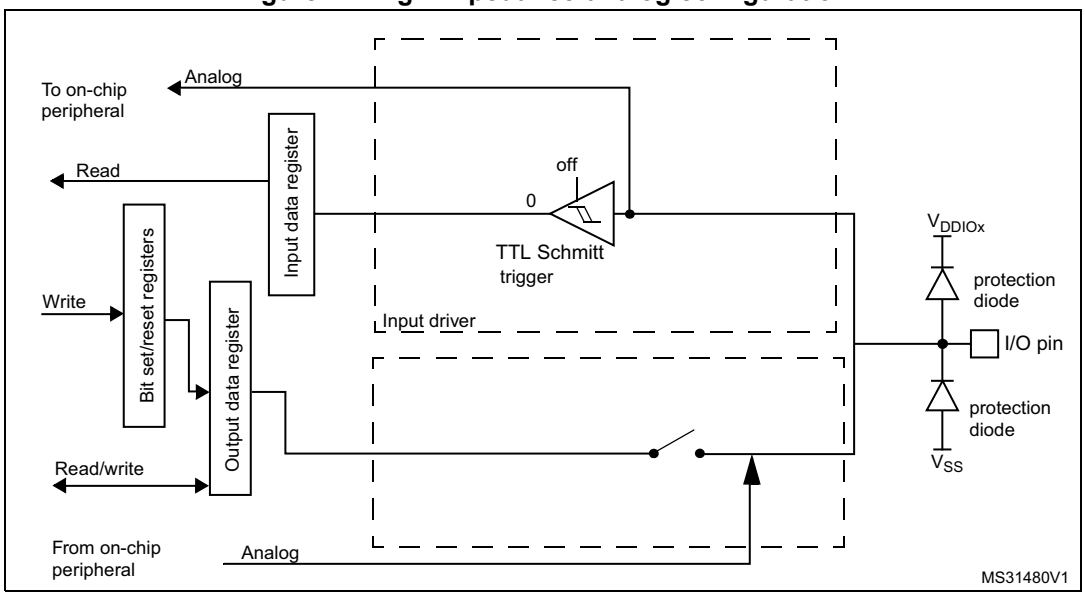
11.3.12 Analog configuration

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 and pull-down resistors are disabled by hardware
- Read access to the input data register gets the value “0”

Figure 42 shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 42. High impedance-analog configuration



11.3.13 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 pin is reserved for clock input and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

11.3.14 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 41.3: RTC functional description*.

11.3.15 Using PH3 as GPIO

PH3 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.

11.4 TrustZone security

The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR. When the TrustZone is active (TZEN=1), each I/O pin of GPIO port can be individually configured as secure through the GPIOx_SECCFGR register.

When the selected I/O pin is configured as secure, its corresponding configuration bits for alternate function, mode selection, I/O data are secure against a non-secure access. In case of non-secure access, these bits are RAZ/WI.

The I/Os with peripherals functions are also conditioned by the peripheral security configuration:

- For peripherals for which the I/O pin selection is done through alternate functions registers: if the peripheral is configured as secure, it cannot be connected to a non-secure I/O pin. If this is not respected, the input data to the secure peripheral is forced to '0' (I/O input pin value is ignored) and the output pin value is forced to '0', thus avoiding any secure information leak through non-secure I/Os.
- For I/Os with analog switches, directly controlled by peripherals (such as ADC for instance): If the I/O is secure, the I/O analog switch cannot be controlled by a non-secure peripheral. If this is not respected, the switch would remain open. This prevent the redirection of secure data to a non-secure peripheral or I/O through analog path.
- Some of the paths between I/Os additional functions mainly and peripherals are not blocked if the I/O is secure and the peripheral is non-secure. Therefore it is recommended to configure those peripherals as secure even when not used by the application. When the path has a security control, it follows the same rule as I/O selection through alternate functions.
- Refer to the device pins definition table in datasheet for more information about peripherals alternate functions and additional functions mapping.

After reset, all GPIO ports are secure.

Table 86 gives a summary of the I/O port secured bits following the security configuration bit in the GPIO_SECCFGR register. When the I/O bit port is configured as secure:

- Secured bits: read and write operations are only allowed by a secure access. Non-secure read or write accesses on secured bits are RAZ/WI. There is no illegal access event generated.
- Non-secure bits: no restriction. Read and write operations are allowed by both secure and non-secure accesses.

When the TrustZone security is disabled (TZEN = 0 in FLASH_OPTR register), all registers bits are non-secure. The GPIOx_SECCFGR register is RAZ/WI.

Table 86. GPIO secured bits

Secure configuration bit	Secured bit	Register name	Non-secure access on secure bits
SECy = 1 in GPIOx_SECCFGR	MODEy[1:0]	GPIOx_MODER	RAZ/WI
	OTy	GPIOx_OTYPER	
	OSPEEDy[1:0]	GPIOx_OSPEEDR	
	PUPDy[1:0]	GPIOx_PUPDR	
	IDy	GPIOx_IDR	
	ODy	GPIOx_ODR	
	BSy	GPIOx_BSRR	
	LCKy	GPIOx_LCKR	
	AFSELy[3:0]	GPIOx_AFRL	
	BRy	GPIOx_AFRH	
		GPIOx_BRR	

Note: GPIOx, x= A..H , and y=0..15

11.5 Privileged and Unprivileged modes

All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or non-secure).

11.6 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 87*.

The peripheral registers can be written in word, half word or byte mode.

11.6.1 GPIO port mode register (GPIOx_MODER) (x =A to H)

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)

Reset value: 0x0000 000F (for port H)

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)

11.6.2 GPIO port output type register (GPIOx_OTYPER) (x = A to H)

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

11.6.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to H)

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(1)
11: Very high speed(1)

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed..

1. Not available for FT_c IOs.

11.6.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to H)

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

11.6.5 GPIO port input data register (GPIOx_IDR) (x = A to H)

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.

11.6.6 GPIO port output data register (GPIOx_ODR) (x = A to H)

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 to H).

11.6.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to H)

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

11.6.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to H)

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

11.6.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to H)

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

11.6.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to H)

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

11.6.11 GPIO port bit reset register (GPIOx_BRR) (x = A to H)

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

11.6.12 GPIO secure configuration register (GPIOx_SECCFGR) (x = A to H)

When the system is secure (TZEN = 1), this register provides write access security and can be written only by a secure access. It is used to configure a selected I/O as secure. A non-secure write access to this register is discarded. A non-secure read is possible, and thus provides visibility on secured I/O pins on the GPIO port.

When the system is not secure (TZEN = 0), this register is WI and its content has no effect.

Address offset: 0x30

Reset value: 0x0000 FFFF (A to G) and 0x0000 000B (H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: Bits are “rw” when TZEN = 1.

Bits 31:16 Reserved

Bits 15:0 **SEC[15:0]**: I/O pin of Port x secure bit enable y (y= 0..15)
These bits are written by software to enable the security I/O port pin.
0: The I/O pin is non-secure
1: The I/O pin is secure. Refer to *Table 86* for all corresponding secured bits.

11.6.13 GPIO register map

The following table gives the GPIO register map and reset values.

Table 87. 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
0x04	**GPIOx_OTYPER** (where x = A..H)	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	**GPIOx_OSPEEDR** (where x = **A to H**)	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
0x10	**GPIOx_IDR** (where x = **A to H**)	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
0x14	**GPIOx_ODR** (where x = **A to H**)	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 H**)	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 H**)	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 H**)	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 H**)	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..H**A to H**))	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
0x30	**GPIOx_SECCFGR** (where x = A...H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value for A...G																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for H																	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

Refer to *Section 2.3 on page 87* for the register boundary addresses.

12 System configuration controller (SYSCFG)

12.1 SYSCFG main features

The STM32L552xx and STM32L552xx devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Managing robustness feature
- Setting SRAM2 write protection and software erase
- Configuring FPU interrupts
- Enabling/disabling the I2C fast-mode plus driving capability on some I/Os and voltage booster for I/Os analog switches
- Configuring TrustZone security register access

12.2 SYSCFG TrustZone security and privilege

SYSCFG TrustZone security

When the TrustZone security is activated, the SYSCFG is able to secure registers from being modified by non-secure accesses.

The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register.

A non-secure read/write access to a secured register is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the SYSCFG illegal access event is enabled in the GTZC_TZIC_IER register.

Privileged/unprivileged mode

The SYSCFG registers can be read and written by privileged and unprivileged accesses except the SYSCFG registers for CPU configuration: SYSCFG_CSLCKR, SYSCFG_FPUIMR and SYSCFG_CNSLCKR registers.

An unprivileged access to a privileged register is RAZ/WI.

Table 88 shows the register security overview.

Table 88. TrustZone security and privilege register accesses

SYSCFG register name	Read/Write access		Privileged /unprivileged access
TrustZone configuration(1)	TZEN=1	TZEN=0	NA
SYSCFG_SECCFGR	Read: no restriction Write: secure access only Write non-secure: is WI and generates an illegal access event	RAZ/WI	No restriction
SYSCFG_CSLCKR	Read/Write: secure access only Read/Write non-secure: is RAZ/WI and generates and an illegal access event	RAZ/WI	Privileged only Unprivileged: RAZ/WI

Table 88. TrustZone security and privilege register accesses (continued)

SYSCFG register name	Read/Write access		Privileged /unprivileged access
TrustZone configuration(1)	**TZEN=1**	**TZEN=0**	**NA**
SYSCFG_FPUIMR	Read/Write secure access only if FPUSEC bit is set Read/Write non-secure: is RAZ/WI and generates an illegal access event	No restriction	Privileged only Unprivileged: RAZ/WI
SYSCFG_CNSLCKR	Read/write: no restriction	No restriction	Privileged only Unprivileged: RAZ/WI
SYSCFG_CFGR1	Read/Write: secure access only for secure bits depending on peripheral security bits in GTZSC_SECFGR register Read/Write non-secure: only for non-secure bits, otherwise is RAZ/WI	No restriction	No restriction
SYSCFG_SWPR, SYSCFG_SWPR2, SYSCFG_SKR, SYSCFG_SCSR	– If SRAM2SEC bit is set: Read/Write: secure access only Read/Write Non-secure: is RAZ/WI and generates an illegal access event – If SRAM2SEC bit is reset: Read/Write: no restriction	No restriction	No restriction
SYSCFG_CFGR2	– If CLASSBSEC bit is set: Read/Write: secure access only Read/Write Non-secure: is RAZ/WI and generates an illegal access event – If CLASSBSEC bit is reset: Read/Write: no restriction	No restriction	No restriction
SYSCFG_RSSCMDR	RAZ/WI if register access is not allowed(2)	RAZ/WI	No restriction

1. TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register.
2. Refer to register description for register access.

12.3 SYSCFG registers

12.3.1 SYSCFG secure configuration register (SYSCFG_SECCFGR)

When the system is secure (TZEN =1), this register provides write access security and can be written only when the access is secure. It can be globally write-protected, or each bit of this register can be individually write-protected.A non-secure write access is WI and generates an illegal access event. There are no read restrictions.

When the system is not secure (TZEN=0), this register is RAZ/WI.

This register can be read and written by privileged and unprivileged access.

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUSEC	SRAM2SEC	CLASSBSEC	SYSCFGSEC
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FPUSEC**: FPU security.
0: SYSCFG_FPUIMR register can be written by secure and non-secure access
1: SYSCFG_FPUIMR register can be written by secure access only.

Bit 2 **SRAM2SEC**: SRAM2 security.
0: SYSCFG_SKR, SYSCFG_SCR and SYSCFG_SWPRx registers can be written by secure and non-secure access
1: SYSCFG_SKR, SYSCFG_SCR and SYSCFG_SWPRx register can be written by secure access only.

Bit 1 **CLASSBSEC**: ClassB security.
0: SYSCFG_CFGR2 register can be written by secure and non-secure access
1: SYSCFG_CFGR2 register can be written by secure access only.

Bit 0 **SYSCFGSEC**: SYSCFG clock control security.
0: SYSCFG configuration clock in RCC registers can be written by secure and non-secure access
1: SYSCFG configuration clock in RCC registers can be written by secure access only.

12.3.2 SYSCFG configuration register 1 (SYSCFG_CFGR1)

When the system is secure (TZEN =1), this register can be a mix of secure and non-secure bits depending on I2Cx, ADC security configuration bit in TZSC_SECCFGR register and GPIO security bits. A non-secure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN=0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

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.	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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ANASWVDD	BOOSTEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

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

Bit 23 **I2C4_FMP**: I2C4 fast-mode plus (Fm+) 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**: 12C3 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**: 12C2 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 I2C2 pins selected through AF selection bits
1: Fm+ mode is enabled on I2C2 pins selected through AF selection bits.

Bit 20 **I2C1_FMP**: 12C1 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 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**: 12C1 fast-mode plus driving capability activation on PB9.
This bit enables the Fm+ driving mode for PB9.
0: PB9 pin operates in standard mode
1: Fm+ mode is enabled on PB9 pin, and the speed control is bypassed.

Bit 18 **I2C_PB8_FMP**: 12C1 fast-mode plus driving capability activation on PB8.
This bit enables the Fm+ driving mode for PB8.
0: PB8 pin operates in standard mode
1: Fm+ mode is enabled on PB8 pin, and the speed control is bypassed.

Bit 17 **I2C_PB7_FMP**: 12C1 fast-mode plus driving capability activation on PB7.
This bit enables the Fm+ driving mode for PB7.
0: PB7 pin operates in standard mode
1: Fm+ mode is enabled on PB7 pin, and the speed control is bypassed.

Bit 16 **I2C_PB6_FMP**: 12C1 fast-mode plus driving capability activation on PB6.
This bit enables the Fm+ driving mode for PB6.
0: PB6 pin operates in standard mode
1: Fm+ mode is 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.
Note: Refer to Table 89: BOOSTEN and ANASWVDD set/reset for bit 9 setting.

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable.
0: I/O analog switches are supplied by VDDA voltage.
This is the recommended configuration when using the ADC in high VDDA voltage operation
1: I/O analog switches are supplied by a dedicated voltage booster (supplied by VDD).
This is the recommended configuration when using the ADC in low VDDA voltage operation.
Note: Refer to Table 89: BOOSTEN and ANASWVDD set/reset for bit 8setting.

Bits 7:0 Reserved, must be kept at reset value.

Table describes when the bit 8 (BOOSTEN) and the bit 9 (ANASWVDD) should be set or reset depending on the voltage settings.

Table 89. BOOSTEN and ANASWVDD set/reset

VDD	VDDA	BOOSTEN	ANASWVDD
-	> 2.4 V	0	0
> 2.4 V	< 2.4 V		1
< 2.4 V		1	0

12.3.3 FPU interrupt mask register (SYSCFG_FPUIMR)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the FPUSEC bit in the SYSCFG_SECCFGR register. A non-secure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN=0), there is no access restriction.

This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

Address offset: 0x08

Reset value: 0x0000 001F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	FPU_IE[5:0]					
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **FPU_IE[5:0]**: Floating point unit interrupts enable bits.
FPU_IE[5]: Inexact interrupt enable (interrupt disable at reset)
FPU_IE[4]: Input abnormal 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

12.3.4 SYSCFG CPU non-secure lock register (SYSCFG_ CNSLCKR)

This register is used to lock the configuration of non-secure MPU and VTOR_NS registers. This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

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.	Res.	Res.	LOCKNSMPU	LOCKNSVTOR
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LOCKNSMPU**: Non-secure MPU registers lock.
This bit is set by software and cleared only by a system reset. When is set, it disables write access to non-secure MPU_CTRL_NS, MPU_RNR_NS and MPU_RBAR_NS registers.
0: Non-secure MPU registers write is enabled
1: Non-secure MPU registers write is disabled.

Bit 0 **LOCKNSVTOR**: VTOR_NS register lock.
This bit is set by software and cleared only by a system reset.
0: VTOR_NS register write is enabled
1: VTOR_NS register write is disabled.

12.3.5 SYSCFG CPU secure lock register (SYSCFG _CSLOCKR)

This register is used to lock the configuration of PRIS and BFHFNMINS bits in the AIRCR register, SAU, secure MPU and VTOR_S registers.

When the system is secure (TZEN =1), this register can be written only when the access is secure. A non-secure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN=0), this register is RAZ/WI

This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKSMPU	LOCKSVTAIRCR
													rs	rs	rs

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKSAU**: SAU registers lock.
This bit is set by software and cleared only by a system reset. When is set, it disables write access to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers.
0: SAU registers write is enabled
1: SAU registers write is disabled.

Bit 1 **LOCKSMPU**: Secure MPU registers lock.
This bit is set by software and cleared only by a system reset. When is set, it disables write access to secure MPU_CTRL, MPU_RNR and MPU_RBAR registers.
0: Secure MPU registers writes is enabled
1: Secure MPU registers writes is disabled.

Bit 0 **LOCKSVTAIRCR**: VTOR_S register and AIRCR register bits lock.
This bit is set by software and cleared only by a system reset. When is set, it disables write access to VTOR_S register, PRIS and BFHFNMINS bits in the AIRCR register.
0: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write is enabled
1: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write is disabled.

12.3.6 SYSCFG configuration register 2 (SYSCFG_CFGR2)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the CLASSBSEC bit in the SYSCFG_SECCFGR register. When CLASSBSEC bit is set, only secure access is allowed. A non-secure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN=0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPF	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL
							rc_w1					rs	rs	rs	rs

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SPF**: SRAM2 parity error flag.
This bit is set by hardware when an SRAM2 parity error is detected. It is cleared by software by writing ‘1’.
0: No SRAM2 parity error detected
1: SRAM2 parity error detected.

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

Bit 3 **ECCL**: ECC lock.
This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Flash ECC error connection to TIM1/8/15/16/17 Break input.
0: ECC error disconnected from TIM1/8/15/16/17 Break input
1: ECC error connected to TIM1/8/15/16/17 Break input.

Bit 2 **PVDL**: PVD lock enable bit.
This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/8/15/16/17 Break input, as well as the PVDE and PLS[2:0] in the PWR_CR2 register.
0: PVD interrupt disconnected from TIM1/8/15/16/17 Break input. PVDE and PLS[2:0] bits can be programmed by the application
1: PVD interrupt connected to TIM1/8/15/16/17 Break input, PVDE and PLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM2 parity lock bit.
This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM2 parity error signal connection to TIM1/8/15/16/17 Break inputs.
0: SRAM2 parity error signal disconnected from TIM1/8/15/16/17 Break inputs
1: SRAM2 parity error signal connected to TIM1/8/15/16/17 Break inputs.

Bit 0 **CLL**: Cortex®-M33 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®-M33 LOCKUP (hardfault) output to TIM1/8/15/16/17 Break input.
0: Cortex®-M33 LOCKUP output disconnected from TIM1/8/15/16/17 Break inputs
1: Cortex®-M33 LOCKUP output connected to TIM1/8/15/16/17 Break inputs.

12.3.7 SYSCFG SRAM2 control and status register (SYSCFG_SCSR)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the SRAM2SEC bit in the SYSCFG_SECCFGR register. When SRAM2SEC bit is set, only secure access is allowed. A non-secure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN=0), here is no access restriction.

This register can be read and written by privileged and unprivileged access.

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.	Res.	Res.	Res.	SRAM2 BSY	SRAM2 ER
														r	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SRAM2BSY**: SRAM2 busy by erase operation.
0: No SRAM2 erase operation is ongoing
1: SRAM2 erase operation is ongoing.

Bit 0 **SRAM2ER**: SRAM2 erase.
Setting this bit starts a hardware SRAM2 erase operation. This bit is automatically cleared at the end of the SRAM2 erase operation

Note: This bit is write-protected: setting this bit is possible only after the correct key sequence is written in the SYSCFG_SKR register.

12.3.8 SYSCFG SRAM2 key register (SYSCFG_SKR)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the SRAM2SEC bit in the SYSCFG_SECCFGR register. When SRAM2SEC bit is set, only secure access is allowed. A non-secure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN=0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w							

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

Bits 7:0 **KEY[7:0]**: SRAM2 write protection key for software erase.
The following steps are required to unlock the write protection of the SRAM2ER bit in the SYSCFG_CFGR2 register.
0: Write “0xCA” into Key[7:0]
1: Write “0x53” into Key[7:0]

Note: Writing a wrong key reactivates the write protection.

12.3.9 SYSCFG SRAM2 write protection register (SYSCFG_SWPR)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the SRAM2SEC bit in the SYSCFG_SECCFGR register. When SRAM2SEC bit is set, only secure access is allowed. A non-secure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN=0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

Address offset: 0x20

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)**: SRAM2 1 Kbyte page x write protection.
These bits are set by software and cleared only by a system reset.
0: Write protection of SRAM2 1 Kbyte page x is disabled
1: Write protection of SRAM2 1 Kbyte page x is enabled.

12.3.10 SYSCFG SRAM2 write protection register 2 (SYSCFG_SWPR2)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the SRAM2SEC bit in the SYSCFG_SECCFGR register. When SRAM2SEC bit is set, only secure access is allowed. A non-secure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN=0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP
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
P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PxWP (x = 32 to 63)**: SRAM2 1 Kbyte page x write protection.
These bits are set by software and cleared only by a system reset.
0: Write protection of SRAM2 1 Kbyte page x is disabled
1: Write protection of SRAM2 1 Kbyte page x is enabled.

12.3.11 SYSCFG RSS command register (SYSCFG_RSSCMDR)

When the system is secure (TZEN =1), this register can be read and written only when the APB access is secure. Otherwise it is RAZ/WI.

When the system is not secure (TZEN=0), this register is RAZ/WI.

This register can be read and written by privileged and unprivileged access.

Address offset: 0x2C

Power-on 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
RSSCMD															
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 **RSSCMD**: RSS commands.

Defines a command to be executed by the RSS.

0x01C0: request boot from RSS with jump to bootloader when BOOT_LOCK bitfield from FLASH_SECBOOTADD0R option byte register and FLASH_OPTR_nBOOT0 bitfield from FLASH_OPTR option byte register are cleared.

12.3.12 SYSCFG register map

The following table gives the SYSCFG register map and the reset values.

Table 90. SYSCFG register map and reset values

Offset	Register name reset value	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_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUSEC	SRAM2SEC	CLASSBSEC	SYSCFGSEC
	Reset value																													0	0	0	0
0x04	SYSCFG_CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	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.	Res.
	Reset value									0	0	0	0	0	0	0	0							0	0								
0x08	SYSCFG_FPUIMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IE[5:0]					
	Reset value																											1	1	1	1	1	1
0x0C	SYSCFG_CNSLCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKNSMPU	LOCKNSVTOR
	Reset value																															0	0
0x10	SYSCFG_CSLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKSMPU	LOCKSVTAIRCR
	Reset value																														0	0	0
0x14	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.	SSPF	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL.
	Reset value																								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.	SRAM2BSY	SRAM2ER
	Reset value																															0	0
0x1C	SYSCFG_SKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY [7:0]							
	Reset value																									0	0	0	0	0	0	0	0
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

Table 90. SYSCFG register map and reset values (continued)

Offset	Register name reset value	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	SYSCFG_SWPR2	P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP	P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
	Reset value	0	0	0	0	0	0	0	0	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	SYSCFG_RSSCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSCMD[15:0]															
	Power-on reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	System reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3 on page 87* for the register boundary addresses.

13 Peripherals interconnect matrix

13.1 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, Stop 1 and Stop 2 modes.

13.2 Connection summary

Table 91. STM32L552xx and STM32L562xx peripherals interconnect matrix(1) (2)

Source	Destination																						
	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17	LPTIM1	LPTIM2	LPTIM3	ADC1	ADC2	DFSDM1	OPAMP1	OPAMP2	DAC1	DAC2	COMP1	COMP2	IRTIM
TIM1	-	*1*	*1*	*1*	*1*	-	-	-	*1*	-	-	-	-	-	*2*	*2*	*5*	-	-	-	-	*9*	-	-
TIM8	-	-	*1*	-	*1*	*1*	-	-	-	-	-	-	-	-	*2*	*2*	*5*	-	-	*4*	*4*	-	*9*	-
TIM2	*1*	*1*	-	*1*	*1*	*1*	-	-	-	-	-	-	-	-	*2*	*2*	-	-	-	*4*	*4*	*9*	-	-
TIM3	*1*	-	*1*	-	*1*	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	*5*	-	-	-	-	*9*	*9*	-
TIM4	*1*	*1*	*1*	*1*	-	*1*	-	-	-	-	-	-	-	-	*2*	*2*	*5*	-	-	*4*	*4*	-	-	-
TIM5	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*4*	*4*	-	-	-
TIM6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*2*	*2*	*5*	-	-	*4*	*4*	-	-	-
TIM7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*5*	-	-	*4*	*4*	-	-	-
TIM15	*1*	-	-	*1*	-	-	-	-	-	-	-	-	-	-	*2*	*2*	-	-	-	-	-	-	*9*	-
TIM16	-	-	-	-	-	-	-	-	*1*	-	-	-	-	-	-	-	*5*	-	-	-	-	-	-	*15*
TIM17	-	-	-	-	-	-	-	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*15*
LPTIM1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*5*	-	-	-	-	-	-	-
LPTIM2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LPTIM3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ADC1	*3*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-	-	-	-	-	-
ADC2	*3*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-	-	-	-	-	-
DFSDM1	*6*	*6*	-	-	-	-	-	-	*6*	*6*	*6*	-	-	-	-	-	-	-	-	-	-	-	-	-
T. Sensor	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*12*	*12*	-	-	-	-	-	-	-	-
VBAT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*12*	*12*	-	-	-	-	-	-	-	-
VREFINT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*12*	*12*	-	-	-	-	-	-	-	-

Table 91. STM32L552xx and STM32L562xx peripherals interconnect matrix(1) (2) (continued)

Source	Destination																							
	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17	LPTIM1	LPTIM2	LPTIM3	ADC1	ADC2	DFSDM1	OPAMP1	OPAMP2	DAC1	DAC2	COMP1	COMP2	IRTIM
OPAMP1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*12*	*12*	-	-	-	-	-	-	-	-
OPAMP2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*12*	*12*	-	-	-	-	-	-	-	-
DAC1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*12*	*12*	-	-	-	-	-
DAC2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HSE	-	-	-	-	-	-	-	-	-	-	*7*	-	-	-	-	-	-	-	-	-	-	-	-	-
LSE	-	-	*7*	-	-	-	-	-	*7*	*7*	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MSI	-	-	-	-	-	-	-	-	-	-	*7*	-	-	-	-	-	-	-	-	-	-	-	-	-
LSI	-	-	-	-	-	-	-	-	-	*7*	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MCO	-	-	-	-	-	-	-	-	-	-	*7*	-	-	-	-	-	-	-	-	-	-	-	-	-
EXTI	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*2*	*2*	*5*	-	-	*4*	*4*	-	-	-
RTC	-	-	-	-	-	-	-	-	-	*7*	-	*8*	*8*	*8*	-	-	-	-	-	-	-	-	-	-
COMP1	*13*	*13*	*13*	*13*	-	-	-	-	*13*	*13*	*13*	*8*	*8*	*8*	-	-	-	-	-	-	-	-	-	-
COMP2	*13*	*13*	*13*	*13*	-	-	-	-	*13*	*13*	*13*	*8*	*8*	*8*	-	-	-	-	-	-	-	-	-	-
SYST ERR	*14*	*14*	-	-	-	-	-	-	*14*	*14*	*14*	-	-	-	-	-	-	-	-	-	-	-	-	-
USB	-	-	*11*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. Numbers in table are links to corresponding detailed sub-section in *Section 13.3: Interconnection details*.
2. The "-" symbol in grayed cells means no interconnect.

13.3 Interconnection details

13.3.1 From timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM15/TIM16/TIM17) to timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM15)

Purpose

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 34.3.19: Timer synchronization*.

The modes of synchronization are detailed in:

- *Section 33.3.26: Timer synchronization* for advanced-control timers (TIM1/TIM8)
- *Section 34.3.18: Timers and external trigger synchronization* for general-purpose timers (TIM2/TIM3/TIM4/TIM5)
- *Section 35.4.19: External trigger synchronization (TIM15 only)* for general-purpose timer (TIM15)

Triggering signals

The output (from Master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1/TIM8) following a configurable timer event.

The input (to slave) is on signals TIMx_ITR0/ITR1/ITR2/ITR3

The input and output signals for TIM1/TIM8 are shown in *Figure 228: Advanced-control timer block diagram*.

The possible master/slave connections are given in:

- *Table 270: TIMx internal trigger connection*
- *Table 275: TIMx internal trigger connection*
- *Table 279: TIMx Internal trigger connection*

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.2 From timer (TIM1/TIM2/TIM3/TIM4/TIM6/TIM8/TIM15) and EXTI to ADC (ADC1/ADC2)

Purpose

General-purpose timers (TIM2/TIM3/TIM4), basic timer (TIM6), advanced-control timers (TIM1/TIM8), general-purpose timer (TIM15) and EXTI can be used to generate an ADC triggering event.

TIMx synchronization is described in: *Section 33.3.27: 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:

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.3 From ADC1/ADC2 to timer (TIM1/TIM8)

Purpose

ADC1/ADC2 can provide trigger event through watchdog signals to advanced-control timers (TIM1/TIM8).

A description of the ADC analog watchdog setting is provided in: .

Trigger settings on the timer are provided in: *Section 33.3.4: External trigger input*.

Triggering signals

The output (from ADC) is on signals ADCn_AWDx_OUT n = 1 (for ADC1,2) x = 1, 2, 3 (3 watchdog per ADC) and the input (to timer) on signal TIMx_ETR (external trigger).

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.4 From timer (TIM2/TIM4/TIM5/TIM6/TIM7/TIM8) and EXTI to DAC (DAC1/DAC2)

Purpose

General-purpose timers (TIM2/TIM4/TIM5), basic timers (TIM6, TIM7), advanced-control timers (TIM8) 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 22.4.6: DAC trigger selection* (single and dual mode).

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.5 From timer (TIM1/TIM3/TIM4/TIM6/TIM7/TIM8/TIM16/LPTIM1) and EXTI to DFSDM1

Purpose

General-purpose timers (TIM3/TIM4), basic timers (TIM6/TIM7), advanced-control timers (TIM1/TIM8), general-purpose timer (TIM16), Low power timer (LPTIM1), EXTI11 and EXTI15 can be used to generate a triggering event on DFSDM1 module (on each possible data block DFSDM1_FLT0/DFSDM1_FLT1/DFSDM1_FLT2/DFSDM1_FLT3) and start an ADC conversion.

DFSDM triggered conversion feature is described in: *Section 26.4.15: Launching conversions*.

Triggering signals

The output (from timer) is on signal TIMx_TRGO/TIMx_TRGO2 or TIM16_OC1.

The input (on DFSDM1) is on signal DFSDM1_INTRG[0:8].

The connection between timers, EXTI and DFSDM1 is provided in *Table 200: DFSDM triggers connection*.

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.6 From DFSDM1 to timer (TIM1/TIM8/TIM15/TIM16/TIM17)

Purpose

DFSDM1 can generate a timer break on advanced-control timers (TIM1/TIM8) and general-purpose timers (TIM15/TIM16/TIM17) when a watchdog is activated (minimum or maximum threshold value crossed by analog signal) or when a short-circuit detection is made.

DFSDM1 watchdog is described in *Section 26.4.10: Analog watchdog*.

DFSDM1 short-circuit detection is described in *Section 26.4.11: Short-circuit detector*.

Timer break is described in:

- *Section 33.3.16: Using the break function* (TIM1/TIM8)
- *Section 35.4.13: Using the break function* (TIM15/TIM16/TIM17)

Triggering signals

The output (from DFSDM1) is on signals dfsdm1_break[0:3] directly connected to timer and 'Ored' with other break input signals of the timer.

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.7 From HSE, LSE, LSI, MSI, MCO, RTC to timer (TIM2/TIM15/TIM16/TIM17)

Purpose

External clocks (HSE, LSE), internal clocks (LSI, MSI), microcontroller output clock (MCO), GPIO and RTC wakeup interrupt can be used as input to general-purpose timer (TIM15/16/17) channel 1.

This allows to calibrate the HSI16/MSI system clocks (with TIM15/TIM16 and LSE) or LSI (with TIM16 and HSE). This is also used to precisely measure LSI (with TIM16 and HSI16) or MSI (with TIM17 and HSI16) oscillator frequency.

When Low Speed External (LSE) oscillator is used, no additional hardware connections are required.

This feature is described in *Section 9.3.18: Internal/external clock measurement with TIM15/TIM16/TIM17*.

External clock LSE can be used as input to general-purpose timers (TIM2) on TIM2_ETR pin, see *Section 34.4.22: TIM2 option register 1 (TIM2_OR1)*.

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.8 From RTC, COMP1, COMP2 to low-power timer (LPTIM1/LPTIM2/LPTIM3)

Purpose

RTC alarm A/B, RTC_TAMP1/2/3 input detection, COMP1/2_OUT can be used as trigger to start LPTIM counters (LPTIM1/2/3).

Triggering signals

This trigger feature is described in *Section 37.4.7: Trigger multiplexer* (and following sections).

The input selection is described in *Table 288: LPTIM1 external trigger connection*.

Active power mode

Run, Sleep, Low-power run, Low-power sleep, Stop 0, Stop 1, Stop 2 (LPTIM1 only).

13.3.9 From timer (TIM1/TIM2/TIM3/TIM8/TIM15) to comparators (COMP1/COMP2)

Purpose

Advanced-control timers (TIM1/TIM8), general-purpose timers (TIM2/TIM3) and general-purpose timer (TIM15) can be used as blanking window input to COMP1/COMP2

The blanking function is described in *Section 24.3.7: Comparator output blanking function*.

The blanking sources are given in:

- *Section 24.6.1: Comparator 1 control and status register (COMP1_CSR)* bits 20:18 BLANKING[2:0]
- *Section 24.6.2: Comparator 2 control and status register (COMP2_CSR)* bits 20:18 BLANKING[2:0]

Triggering signals

Timer output signal TIMx_Ocx are the inputs to blanking source of COMP1/COMP2.

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.10 From ADC (ADC1) to ADC (ADC2)

Purpose

ADC1 can be used as a “master” to trigger ADC2 “slave” start of conversion.

In dual ADC mode, the converted data of the master and slave ADCs can be read in parallel.

Triggering signals

Internal to the ADCs.

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.11 From USB to timer (TIM2)

Purpose

USB (FS SOF) can generate a trigger to general-purpose timer (TIM2).

Connection of USB to TIM2 is described in *Table 275: TIMx internal trigger connection*.

Triggering signals

Internal signal generated by USB FS Start Of Frame.

Active power mode

Run, Sleep.

13.3.12 From internal analog source to ADC (ADC1/ADC2) and OPAMP (OPAMP1/OPAM2)

Purpose

Internal temperature sensor (V_{TS}) and V_{BAT} monitoring channel are connected to ADC1 input channels.

Internal reference voltage (V_{REFINT}) is connected to ADC1 input channels.

OPAMP1 and OPAMP2 outputs can be connected to ADC1 or ADC2 input channels through the GPIO.

DAC1_OUT1 can be connected to OPAMP1_VINP.

DAC1_OUT2 can be connected to OPAMP2_VINP.

This is according:

- *Section 21.2: ADC main features*
- *Section 21.4.11: Channel selection (SQRx, JSQRx)*
- *Section Figure 93.: ADC1 connectivity*
- *Table 193: Operational amplifier possible connections*

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.13 From comparators (COMP1/COMP2) to timers (TIM1/TIM2/TIM3/TIM8/TIM15/TIM16/TIM17)

Purpose

Comparators (COMP1/COMP2) output values can be connected to timers (TIM1/TIM2/TIM3/TIM8/TIM15/TIM16/TIM17) input captures or TIMx_ETR signals.

The connection to ETR is described in *Section 33.3.4: External trigger input*.

Comparators (COMP1/COMP2) output values can also generate break input signals for timers (TIM1/TIM8) on input pins TIMx_BKIN or TIMx_BKIN2 through GPIO alternate function selection using open drain connection of IO, see *Section 33.3.17: Bidirectional break inputs*.

The possible connections are given in:

- *Section 33.4.23: TIM1 option register 1 (TIM1_OR1)*
- *Section 33.4.24: TIM8 option register 1 (TIM8_OR1)*
- *Section 34.4.22: TIM2 option register 1 (TIM2_OR1)*
- *Section 34.4.23: TIM3 option register 1 (TIM3_OR1)*
- *Section 34.4.24: TIM2 option register 2 (TIM2_OR2)*
- *Section 34.4.25: TIM3 option register 2 (TIM3_OR2)*
- *Section 35.3: TIM16/TIM17 main features*

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.14 From system errors to timers (TIM1/TIM8/TIM15/TIM16/TIM17)

Purpose

CSS, CPU hardfault, RAM parity error, FLASH ECC double error detection, PVD can generate system errors in the form of timer break toward timers (TIM1/TIM8/TIM15/TIM16/TIM17).

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

List of possible source of break are described in:

- *Section 33.3.16: Using the break function* (TIM1/TIM8)
- *Section 35.4.13: Using the break function* (TIM15/TIM16/TIM17)
- *Figure 339: TIM15 block diagram*
- *Figure 340: TIM16/TIM17 block diagram*

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.15 From timers (TIM16/TIM17) to IRTIM

Purpose

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 38: Infrared interface (IRTIM)*.

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

13.3.16 From ADC (ADC1/ADC2) to DFSDM

Purpose

Up to 3 internal ADC results can be directly connected through a parallel bus to DFSDM input in order to use DFSDM filtering capabilities.

The feature is described as part of DFSDM peripheral description in *Section 26.4.6: Parallel data inputs - Input from internal ADC*

The possible connections are given in:

- *Section 26.7.1: DFSDM channel y configuration register (DFSDM_CHyCFGR1)*
  - Bits 13:12 DATMPX[1:0]: Input data multiplexer for channel y
- *Section 26.7.5: DFSDM channel y data input register (DFSDM_CHyDATINR)*
  - Bits 31:16 INDAT0[15:0]: Input data for channel y or channel y+1
  - Bits 15:0 INDAT0[15:0]: Input data for channel y

Active power mode

Run, Sleep, Low-power run, Low-power sleep.

14 Direct memory access controller (DMA)

14.1 Introduction

The direct memory access (DMA) controller is a bus master and system peripheral.

The DMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.

The DMA controller features a single AHB master architecture.

There are two instances of DMA, DMA1 and DMA2.

Each channel is dedicated to managing memory access requests from one or more peripherals. Each DMA includes an arbiter for handling the priority between DMA requests.

14.2 DMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access, as source and destination, to on-chip memory-mapped devices such as Flash memory, SRAM, and AHB and APB peripherals
- All DMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (4 levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to 2^{18} - 1
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.
- TrustZone support:
  - Support for AHB secure and non-secure DMA transfers, independently of a first channel level, and independently at a source and destination sub-level
  - TrustZone-aware AHB slave port, protecting any secure resource (register, register field) from a non-secure software access
- Privileged/unprivileged support:
  - Support for AHB privileged and unprivileged DMA transfers, independently of a channel level
  - Privileged-aware AHB slave port

14.3 DMA implementation

14.3.1 DMA1 and DMA2

DMA1 and DMA2 are implemented with the hardware configuration parameters shown in *Table 92*.

Table 92. DMA1 and DMA2 implementation

Feature	DMA1	DMA2
Number of channels (double-buffer)	8	8
TrustZone	1 (supported)	1 (supported)

14.3.2 DMA request mapping

The DMA controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX peripheral.

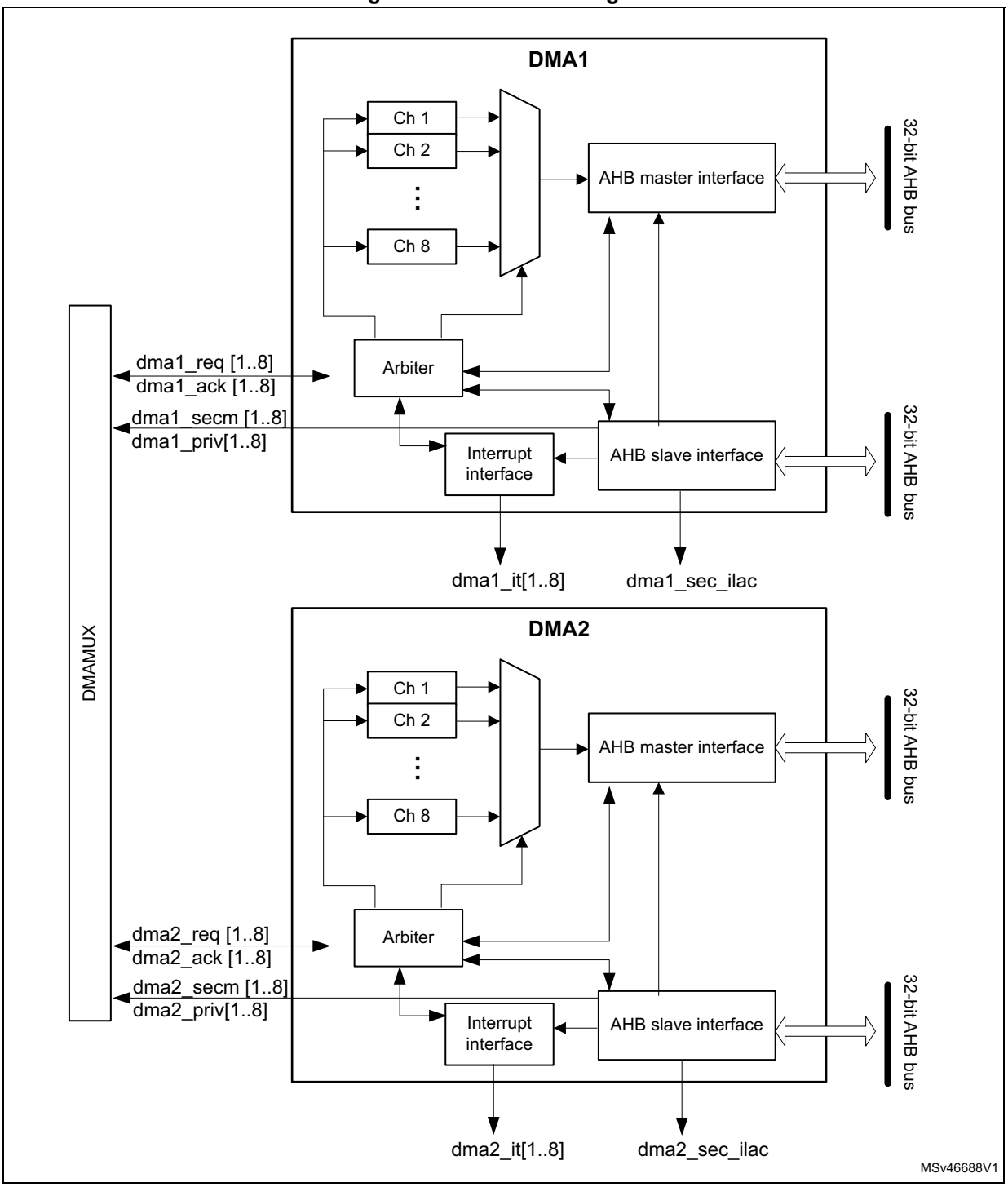
For the mapping of the different requests, refer to the *Section 15.3: DMAMUX implementation*.

14.4 DMA functional description

14.4.1 DMA block diagram

The DMA block diagram is shown in *Figure 43*.

Figure 43. DMA block diagram



The DMA controller performs direct memory transfer by sharing the AHB system bus with other system masters. The bus matrix implements round-robin scheduling. DMA requests may stop the CPU access to the system bus for a number of bus cycles, when CPU and DMA target the same destination (memory or peripheral).

According to its configuration through the AHB slave interface, the DMA controller arbitrates between the DMA channels and their associated received requests. The DMA controller also schedules the DMA data transfers over the single AHB port master.

The DMA controller generates a secure bus and a privileged bus, to keep the DMAMUX peripheral informed of the secure or non-secure state, and the privileged or unprivileged state of each channel x.

The DMA controller generates an interrupt per channel to the interrupt controller.

The DMA controller also generates an illegal access event, as a pulse, to the TrustZone interrupt controller, when a non-secure software attempts to access a secure DMA register or register field.

14.4.2 DMA pins and internal signals

Table 93. DMA internal input/output signals

Signal name	Signal type	Description
dma_req[x]	Input	DMA channel x request
dma_ack[x]	Output	DMA channel x acknowledge
dma_it[x]	Output	DMA channel x interrupt
dma_secm[x]	Output	DMA channel x secure state
dma_priv[x]	Output	DMA channel x privileged state
dma_sec_ilac	Output	DMA global secure illegal access event

14.4.3 DMA transfers

The secure software configures the DMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB secure or non-secure, privileged or unprivileged bus transfers.

A DMA block transfer may be requested from a peripheral, or triggered by the software in case of memory-to-memory transfer.

After an event, the following steps of a single DMA transfer occur:

1. The peripheral sends a single DMA request signal to the DMA controller.
2. The DMA controller serves the request, depending on the priority of the channel associated to this peripheral request.
3. As soon as the DMA controller grants the peripheral, an acknowledge is sent to the peripheral by the DMA controller.
4. The peripheral releases its request as soon as it gets the acknowledge from the DMA controller.
5. Once the request is de-asserted by the peripheral, the DMA controller releases the acknowledge.

The peripheral may order a further single request and initiate another single DMA transfer.

The request/acknowledge protocol is used when a peripheral is either the source or the destination of the transfer. For example, in case of memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the DMA controller. The DMA controller reads then a single data in the memory and writes this data to the peripheral.

For a given channel x, a DMA block transfer consists of a repeated sequence of:

- a single DMA transfer, encapsulating two AHB transfers of a single data, over the DMA AHB bus master:
  - a single data read (byte, half-word or word) from the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
    The start address used for the first single transfer is the base address of the peripheral or memory, and is programmed in the DMA_CPARx or DMA_CM0/1ARx 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_CM0/1ARx 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.

14.4.4 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.

14.4.5 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.

Each channel is either secure or non-secure.

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_CM0/1ARx 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_CM0ARx) 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_CM0/1ARx registers.

Security

The DMA controller is compliant with the TrustZone-M hardware architecture, partitioning all its resources so that they exist in one of the two worlds: the secure world and the non-secure world, at any given time.

A secure software is able to access any resource/register, whatever secure or non-secure. A non-secure software is restricted to access any non-secure resource/register.

Any channel is in a secure or non-secure state, as securely configured by the DMA_CCRx.SECM secure register bit.

When a channel x is configured in secure mode, the following access controls rules are applied:

- A non-secure read access to a register field of this channel is forced to return 0, except for both the secure state and the privileged state of this channel x (SECM and PRIV bits of the DMA_CCRx register) which are readable by a non-secure software.
- A non-secure write access to a register field of this channel has no impact.

When a channel is configured in secure mode, a secure software can separately configure as secure or non-secure the AHB DMA master transfer from the source (by the DMA_CCRx.SSEC register bit), and as secure or non-secure the AHB DMA master transfer to the destination (by the DMA_CCRx.DSEC register bit).

The DMA controller generates a secure bus, dma_secm[7:0], reflecting the DMA_CCRx.SECM register, in order to keep the other hardware peripherals like the DMAMUX, informed of the secure/non-secure state of each DMA channel x.

The DMA controller also generates a security illegal access pulse event, dma_sec_ilac, on an illegal non-secure software access to a secure DMA register or register field. This event is routed to the TrustZone interrupt controller.

The dma_sec_ilac event is generated in the configurations described below:

- If the channel x is in secure state (SECM bit of the DMA_CCRx register set), the dma_sec_ilac is generated on one of the following accesses:
  - a non-secure write access to a dedicated register of this channel x (DMA_CCRx, DMA_CNDTRx, DMA_CPARx, DMA_CM0ARx and DMA_CM1ARx)
  - a non-secure read access to a dedicated register of this channel x, except the DMA_CxCR register (DMA_CNDTRx, DMA_CPARx, DMA_CM0ARx, and DMA_CM1ARx).
- If the channel x is in non-secure state (SECM bit of the DMA_CCRx register cleared), the dma_sec_ilac is generated on a non-secure write access to the DMA_CCRx register which attempts to write 1 into any of the secure configuration bits SECM, DSEC, SSEC.

When the software is switching from a secure state to a non-secure state (after the secure transfer is completed), the secure software must disable the channel by a 32-bit write at the DMA_CCRx address before switching. This operation is needed for the two below reasons:

- a non-secure software cannot do so
- the EN bit of the DMA_CCRx register must be cleared before the (non-secure) software can reprogram the DMA_CCRx for a next transfer.

Note: A trusted application may require that the secure software does not only disable the channel, but also reset the full DMA_CCRx word register to its reset value, as well as reset any other DMA register corresponding to this channel x.

Privileged / unprivileged mode

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged software via the PRIV bit of the DMA_CCRx register.

When a channel x is configured in privileged mode, the following access controls rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for both the privileged state and the secure state of this channel x (PRIV and

SECM bits of the DMA_CCRx register) which are readable by an unprivileged software.

- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) mode, the AHB master transfers from the source and to the destination, are privileged (respectively unprivileged).

DMA generates a privileged bus, dma_priv[7:0], reflecting the PRIV bit of the DMA_CCRx register, in order to keep the other hardware peripherals, like DMAMUX, informed of the privileged / unprivileged state of each DMA channel x.

Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

1. Set a channel x to secure or non-secure, by a secure write access to the secure SECM bit of the DMA_CCRx register. Set a channel x to privileged or unprivileged, by a privileged write access to the privileged PRIV bit of the DMA_CCRx register.
2. Set the 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.
3. Set the memory address in the DMA_CM0ARx 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.
4. Configure the total number of data to transfer in the DMA_CNDTRx register.
   After each data transfer, this value is decremented.
5. Configure the parameters listed below in the DMA_CCRx register:
   – the channel priority
   – the data transfer direction
   – the security level of the data transfers from source and to destination when the channel is secure
   – the circular mode
   – the double-buffer 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
6. 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.

Double-buffer mode (in memory-to-peripheral and peripheral-to memory transfers)

The DMA channels can operate in double-buffer mode.

The difference compared to a regular operation is that the DMA controller toggles between two memory address pointers at the end of each DMA transfer, thus accessing two memory areas in an alternate way. This allows the software to access one of the two memory areas while the DMA controller accesses the other one. The double-buffer mode transfer operates in both directions, so the target memory can be either the source or the destination.

The double-buffer mode is configured by setting both the DBM and CIRC bits of the DMA_CCRx register.

Note: The double-buffer mode must not be used in memory-to-memory mode. Before enabling a channel in double-buffer mode (DBM = 1), the software has to configure appropriately the MEM2MEM bit (MEM2MEM = 0).

The steps described below allow the configuration of a DMA channel x in double-buffer mode:

- Set the DBM and CIRC bits and clear the MEM2MEM bit of the DMA_CCRx register. The circular mode is then activated for the swap mechanism to occur.
- Configure the second memory address register DMA_CM1ARx.
- Continue with the regular channel configuration procedure, and lastly, activate the channel by setting the EN bit of the DMA_CCRx register. The first DMA transfer target memory of the corresponding DMA channel x, is given by the CT bit of the DMA_CCRx register.

Note: Independently from the value of DBM bit of the DMA_CCRx register, if CT = 1, the memory address pointer for the DMA transfer is defined by DMA_CM1ARx, and not by DMA_CM0ARx.

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.

14.4.6 DMA data width, alignment and endianness

When PSIZE[1:0] and MSIZE[1:0] are not equal, the DMA controller performs some data alignments as described in *Table 94*.

Table 94. 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_CM0/1ARx if DIR = 1, else DMA_CPARx)	DMA transfers	Destination content: address / data (DMA_CPARx if DIR = 1, else DMA_CM0/1ARx)
8	8	8	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write B0[7:0] @0x0 2: read B1[7:0] @0x1 then write B1[7:0] @0x1 3: read B2[7:0] @0x2 then write B2[7:0] @0x2 4: read B3[7:0] @0x3 then write B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 00B0[15:0] @0x0 2: read B1[7:0] @0x1 then write 00B1[15:0] @0x2 3: read B2[7:0] @0x2 then write 00B2[15:0] @0x4 4: read B3[7:0] @0x3 then write 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 000000B0[31:0] @0x0 2: read B1[7:0] @0x1 then write 000000B1[31:0] @0x4 3: read B2[7:0] @0x2 then write 000000B2[31:0] @0x8 4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0 2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1 3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2 4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6

Table 94. Programmable data width and endian behavior (when PINC = MINC = 1) (continued)

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_CM0/1ARx if DIR = 1, else DMA_CPARx)	DMA transfers	Destination content: address / data (DMA_CPARx if DIR = 1, else DMA_CM0/1ARx)
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.

14.4.7 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.

Additionally, a security illegal access pulse signal is generated on an illegal non-secure software access to a secure DMA register. This signal is routed to the TrustZone interrupt controller.

14.5 DMA interrupts

An interrupt can be generated on a half transfer, transfer complete or transfer error for each DMA channel x (whatever the channel is secure or non-secure). Separate interrupt enable bits are available for flexibility.

Table 95. 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	-

14.6 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).

14.6.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

This register may mix secure and non secure information, depending on the secure mode of each channel (SECM bit of the DMA_CCRx register). A secure software can read the full interrupt status. A non-secure software is restricted to read the status of non-secure channel(s), other secure bit fields returning zero.

This register may mix privileged and unprivileged information, depending on the privileged mode of each channel (PRIV bit of the DMA_CCRx register). A privileged software can read

the full interrupt status. An unprivileged software is restricted to read the status of unprivileged channel(s), other privileged bit fields returning zero.

Every status / flag bit is set by hardware, independently of the privileged and the secure mode of the channel.

Every status bit is cleared by hardware when the software sets the corresponding clear bit or the corresponding global clear bit CGIFx, in the DMA_IFCR register, provided that, if the channel x is in privileged mode and/or in secure mode, then the software access to DMA_IFCR is also privileged and/or secure.

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

14.6.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

This register may mix secure and non secure information, depending on the secure mode of each channel (SECM bit of the DMA_CCRx register).

A secure software is able to set any flag clear bit of the DMA_IFCR, and order DMA hardware to clear any corresponding flag(s) in the DMA_ISR register.

A non-secure software is restricted to order DMA hardware to clear the non-secure flag(s) in the DMA_ISR, by setting any non-secure corresponding flag clear bit(s) of the DMA_IFCR register.

This register may mix privileged and unprivileged information, depending on the privileged mode of each channel (PRIV bit of the DMA_CCRx register).

A privileged software is able to set any flag clear bit of the DMA_IFCR, and order DMA hardware to clear any corresponding flag(s) in the DMA_ISR register.

An unprivileged software is restricted to order DMA hardware to clear the unprivileged flag(s) in the DMA_ISR, by setting any unprivileged corresponding flag clear bit(s) of the DMA_IFCR register.

Setting the global clear bit CGIFx of the channel x in this DMA_IFCR register, causes the DMA hardware to clear the corresponding GIFx bit and any individual flag among TEIFx, HTIFx, TCIFx, in the DMA_ISR register.

Setting any individual clear bit among CTEIFx, CHTIFx, CTCIFx in this DMA_IFCR register, causes the DMA hardware to clear the corresponding individual flag and the global flag GIFx in the DMA_ISR register, provided that none of the two other individual flags is set.

Writing 0 into any flag clear bit has no effect.

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

14.6.3 DMA channel x configuration register (DMA_CCRx)

Address offset: 0x08 + 0x14 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

This register contains secure and privileged information: the secure state and the privileged state of the channel x (SECM and PRIV control bits).

Modifying the SECM bit must be performed by a secure write access to this register.
Modifying the PRIV bit must be performed by a privileged write access to this register.

Setting any of the DSEC or SSEC bits must be performed by a secure write access to this register.

Except SECM and PRIV control bits, any other register field is non-readable by a non-secure software if the SECM bit is set, and non-readable by an unprivileged software if the PRIV bit is set.

The register fields/bits PRIV, DSEC, SSEC, SECM, CT, DBM, MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	DSEC	SSEC	SECM	CT
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBM	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	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **PRIV**: privileged mode

This bit can only be set and cleared by a privileged software.

0: disabled

1: enabled

This bit must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 19 **DSEC**: security of the DMA transfer to the destination

This bit can only be read, set or cleared by a secure software. It must be a privileged software if the channel is in privileged mode.

This bit is cleared by hardware when the securely written data bit 17 is cleared (on a secure reconfiguration of the channel as non -secure).

A non-secure read to this secure configuration bit returns 0.

A non-secure write of 1 to this secure configuration bit has no impact on the register setting and an illegal access pulse is asserted.

Destination (peripheral or memory) of the DMA transfer is defined by the direction DIR configuration bit.

0: non-secure DMA transfer to the destination

1: secure DMA transfer to the destination

This bit must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 18 **SSEC**: security of the DMA transfer from the source

This bit can only be accessed - read, set or cleared - by a secure software. It must be a privileged software if the channel is in privileged mode.

This bit is cleared by hardware when the securely written data bit 17 is cleared (on a secure reconfiguration of the channel as non -secure).

A non-secure read to this secure configuration bit returns 0.

A non-secure write of 1 to this secure configuration bit has no impact on the register setting and an illegal access pulse is asserted.

Source (peripheral or memory) of the DMA transfer is defined by the direction DIR configuration bit.

0: non-secure DMA transfer from the source

1: secure DMA transfer from the source

This bit must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 17 **SECM**: secure mode

This bit can only be set or cleared by a secure software.

0: non-secure channel

1: secure channel

This bit must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 16 **CT**: current target memory of DMA transfer in double-buffer mode

This bit is toggled by hardware at the end of each channel transfer in double-buffer mode.

0: memory 0 (addressed by the DMA_CM0AR pointer)

1: memory 1 (addressed by the DMA_CM1AR pointer)

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 15 **DBM**: double-buffer mode

This bit must be set only in memory-to-peripheral and peripheral-to-memory transfers (MEM2MEM=0). The CIRC bit must also be set in double buffer mode.

0: disabled (no memory address switch at the end of the DMA transfer)

1: enabled (memory address switched at the end of the DMA transfer)

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 14 **MEM2MEM**: memory-to-memory mode

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 13:12 **PL[1:0]**: priority level

00: low

01: medium

10: high

11: very high

Note: This field is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 11:10 **MSIZE[1:0]**: memory size

Defines the data size of each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

Note: This field is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 9:8 **PSIZE[1:0]**: peripheral size

Defines the data size of each DMA transfer to the identified peripheral.

In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

Note: This field is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 7 **MINC**: memory increment mode

Defines the increment mode for each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 6 **PINC**: peripheral increment mode

Defines the increment mode for each DMA transfer to the identified peripheral.

n memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 5 **CIRC**: circular mode

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 4 **DIR**: data transfer direction

This bit must be set only in memory-to-peripheral and peripheral-to-memory modes.

0: read from peripheral

- Source attributes are defined by PSIZE and PINC, plus the DMA_CPARx register. This is still valid in a memory-to-memory mode.
- Destination attributes are defined by MSIZE and MINC, plus the DMA_CM0/1ARx 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_CM0/1ARx register. This is still valid in a peripheral-to-peripheral mode.

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 3 **TEIE**: transfer error interrupt enable

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 2 **HTIE**: half transfer interrupt enable

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 1 **TCIE**: transfer complete interrupt enable

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 0 **EN**: channel enable

When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

0: disabled

1: enabled

Note: This bit is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode)

14.6.4 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.	NDT[17:16]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **NDT[17:0]**: number of data to transfer (0 to 2^{18} - 1)

This field is updated by hardware when the channel is enabled:

- It is decremented after each single DMA 'read followed by write' transfer, indicating the remaining amount of data items to transfer.
- It is kept at zero when the programmed amount of data to transfer is reached, if the channel is not in circular mode (CIRC = 0 in the DMA_CCRx register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this field is zero, no transfer can be served whatever the channel status (enabled or not).

Note: This field is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

14.6.5 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 is read/written.

When PSIZE[1:0] = 01 (16 bits), bit 0 of PA[31:0] is ignored. Access is automatically aligned to a half-word address.

When PSIZE = 10 (32 bits), bits 1 and 0 of PA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory destination address if DIR = 1 and the memory source address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral destination address DIR = 1 and the peripheral source address if DIR = 0.

Note: This register is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

14.6.6 DMA channel x memory 0 address register (DMA_CM0ARx)

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 is read/written.

When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.

When MSIZE = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral source address DIR = 1 and the peripheral destination address if DIR = 0.

Note: This register is set and cleared by software (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

14.6.7 DMA channel x memory 1 address register (DMA_CM1ARx)

Address offset: 0x18 + 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 (privileged/secure software if the channel is in privileged/secure mode).
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

14.6.8 DMA register map

Table 96. 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 96. 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.	PRIV	DSEC	SSEC	SECM	CT	DBM	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	0	0	0	0	0
0x00C	DMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CM0AR1	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	DMA_CM1AR1	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
0x01C	DMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	DSEC	SSEC	SECM	CT	DBM	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	0	0	0	0	0
0x020	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CM0AR2	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	DMA_CM1AR2	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
0x030	DMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	DSEC	SSEC	SECM	CT	DBM	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	0	0	0	0	0
0x034	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_CM0AR3	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	DMA_CM1AR3	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
0x044	DMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	DSEC	SSEC	SECM	CT	DBM	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	0	0	0	0	0
0x048	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	DMA_CPAR4	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 96. 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
0x050	DMA_CM0AR4	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	DMA_CM1AR4	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
0x058	DMA_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	DSEC	SSEC	SECM	CT	DBM	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	0	0	0	0	0
0x05C	DMA_CNDTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DMA_CPAR5	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	DMA_CM0AR5	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	DMA_CM1AR5	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
0x06C	DMA_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	DSEC	SSEC	SECM	CT	DBM	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	0	0	0	0	0
0x070	DMA_CNDTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	DMA_CPAR6	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	DMA_CM0AR6	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	DMA_CM1AR6	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
0x080	DMA_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	DSEC	SSEC	SECM	CT	DBM	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	0	0	0	0	0
0x084	DMA_CNDTR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	DMA_CPAR7	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	DMA_CM0AR7	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	DMA_CM1AR7	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
0x094	DMA_CCR8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	DSEC	SSEC	SECM	CT	DBM	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	0	0	0	0	0
0x098	DMA_CNDTR8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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

Table 96. 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
0x0A0	DMA_CM0AR8	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
0x0A4	DMA_CM1AR8	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 2.3* for the register boundary addresses.

15 DMA request multiplexer (DMAMUX)

15.1 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 15.3.1*.

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 15.3.2*.

15.2 DMAMUX main features

- 16-channel programmable DMA request line multiplexer output
- 4-channel DMA request generator
- 23 trigger inputs to DMA request generator
- 23 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:
  - 90 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
- TrustZone support:
  - Support for AHB secure and non-secure DMA transfers, independently at a channel level.
  - TrustZone-aware AHB slave port, protecting any secure resource (register, register field) from a non-secure software access, with configurable interrupt event.
  - Two secure and non-secure interrupt requests, resulting from any of the respectively secure and non-secure channels. Each channel event being caused from any of the two DMAMUX input events: trigger or synchronization overrun, associated with a respectively secure and non-secure channels.
- Privileged / Unprivileged support:
  - Support for AHB privileged and unprivileged DMA transfers, independently, at a channel level.
  - Privileged-aware AHB slave port.

15.3 DMAMUX implementation

15.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

Table 97. DMAMUX instantiation

Feature	DMAMUX
Number of DMAMUX output request channels	16
Number of DMAMUX request generator channels	4
Number of DMAMUX request trigger inputs	23

Table 97. DMAMUX instantiation (continued)

Feature	DMAMUX
Number of DMAMUX synchronization inputs	23
Number of DMAMUX peripheral request inputs	90
DMAMUX TrustZone support	1

15.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

DMAMUX is used with DMA1 and DMA2:

- DMAMUX channels 0 to 7 are connected to DMA1 channels 0 to 7
- DMAMUX channels 8 to 15 are connected to DMA2 channels 0 to 7

Table 98. DMAMUX: assignment of multiplexer inputs to resources

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux_req_gen0	44	TIM1_CH3	87	DFSDM1_FLT1
2	dmamux_req_gen1	45	TIM1_CH4	88	DFSDM1_FLT2
3	dmamux_req_gen2	46	TIM1_UP	89	DFSDM1_FLT3
4	dmamux_req_gen3	47	TIM1_TRIG	90	AES_IN
5	ADC1	48	TIM1_COM	91	AES_OUT
6	ADC2	49	TIM8_CH1	92	HASH_IN
7	DAC1	50	TIM8_CH2	93	USBPD_TX
8	DAC2	51	TIM8_CH3	94	USBPD_RX
9	TIM6_UP	52	TIM8_CH4	95	Reserved
10	TIM7_UP	53	TIM8_UP	96	Reserved
11	SPI1_RX	54	TIM8_TRIG	97	Reserved
12	SPI1_TX	55	TIM8_COM	98	Reserved
13	SPI2_RX	56	TIM2_CH1	99	Reserved
14	SPI2_TX	57	TIM2_CH2	100	Reserved
15	SPI3_RX	58	TIM2_CH3	101	Reserved
16	SPI3_TX	59	TIM2_CH4	102	Reserved
17	I2C1_RX	60	TIM2_UP	103	Reserved
18	I2C1_TX	61	TIM3_CH1	104	Reserved
19	I2C2_RX	62	TIM3_CH2	105	Reserved
20	I2C2_TX	63	TIM3_CH3	106	Reserved
21	I2C3_RX	64	TIM3_CH4	107	Reserved
22	I2C3_TX	65	TIM3_UP	108	Reserved
23	I2C4_RX	66	TIM3_TRIG	109	Reserved

Table 98. DMAMUX: assignment of multiplexer inputs to resources (continued)

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
24	I2C4_TX	67	TIM4_CH1	110	Reserved
25	USART1_RX	68	TIM4_CH2	111	Reserved
26	USART1_TX	69	TIM4_CH3	112	Reserved
27	USART2_RX	70	TIM4_CH4	113	Reserved
28	USART2_TX	71	TIM4_UP	114	Reserved
29	USART3_RX	72	TIM5_CH1	115	Reserved
30	USART3_TX	73	TIM5_CH2	116	Reserved
31	UART4_RX	74	TIM5_CH3	117	Reserved
32	UART4_TX	75	TIM5_CH4	118	Reserved
33	UART5_RX	76	TIM5_UP	119	Reserved
34	UART5_TX	77	TIM5_TRIG	120	Reserved
35	LPUART1_RX	78	TIM15_CH1	121	Reserved
36	LPUART1_TX	79	TIM15_UP	122	Reserved
37	SAI1_A	80	TIM15_TRIG	123	Reserved
38	SAI1_B	81	TIM15_COM	124	Reserved
39	SAI2_A	82	TIM16_CH1	125	Reserved
40	SAI2_B	83	TIM16_UP	126	Reserved
41	OCTOSPI1	84	TIM17_CH1	127	Reserved
42	TIM1_CH1	85	TIM17_UP	-	-
43	TIM1_CH2	86	DFSDM1_FLT0	-	-

Table 99. DMAMUX: assignment of trigger inputs to resources

Trigger input	Resource	Trigger input	Resource
0	EXTI LINE0	16	dmamux_evt0
1	EXTI LINE1	17	dmamux_evt1
2	EXTI LINE2	18	dmamux_evt2
3	EXTI LINE3	19	dmamux_evt3
4	EXTI LINE4	20	LPTIM1_OUT
5	EXTI LINE5	21	LPTIM2_OUT
6	EXTI LINE6	22	LPTIM3_OUT
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

Table 99. DMAMUX: assignment of trigger inputs to resources (continued)

Trigger input	Resource	Trigger input	Resource
13	EXTI LINE13	29	Reserved
14	EXTI LINE14	30	Reserved
15	EXTI LINE15	31	Reserved

Table 100. DMAMUX: assignment of synchronization inputs to resources

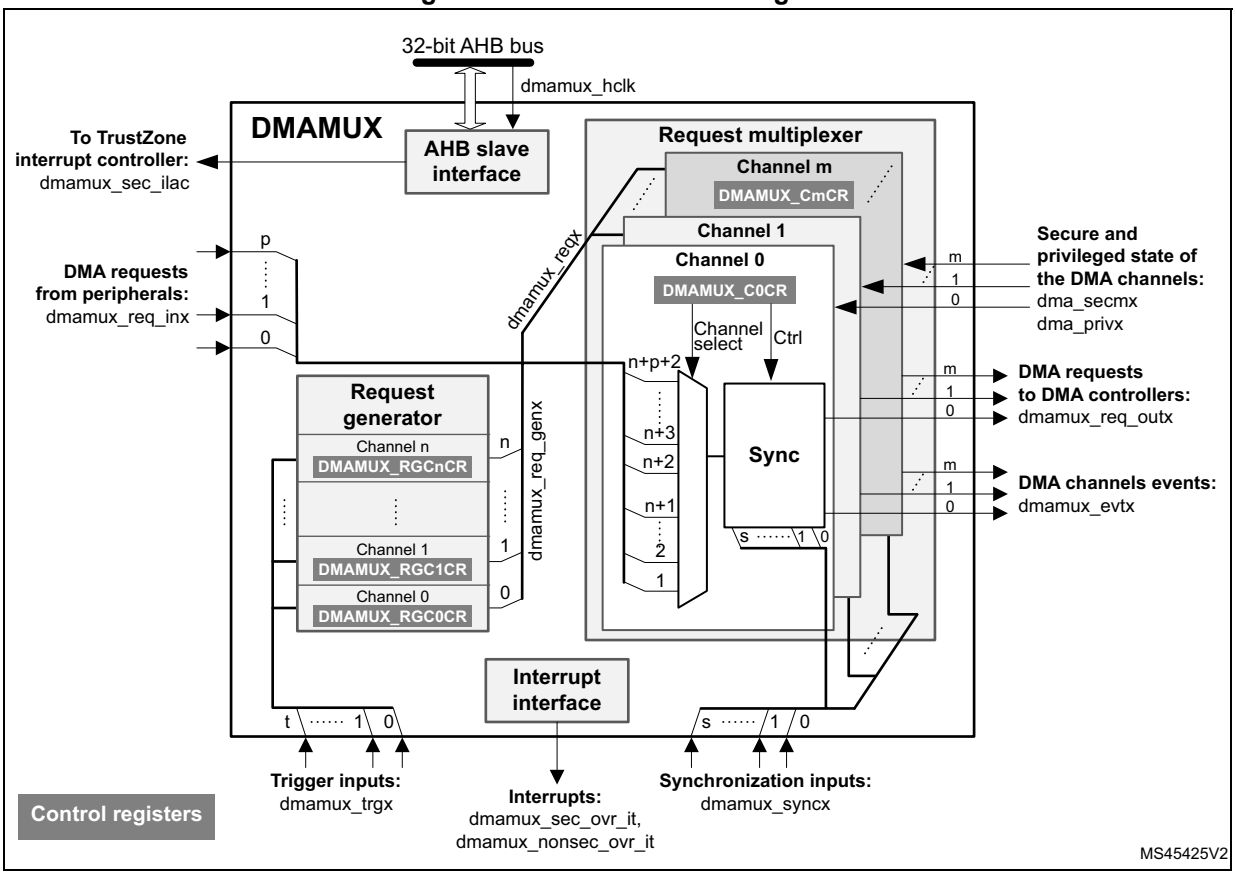
Sync. input	Resource	Sync. input	Resource
0	EXTI LINE0	16	dmamux_evt0
1	EXTI LINE1	17	dmamux_evt1
2	EXTI LINE2	18	dmamux_evt2
3	EXTI LINE3	19	dmamux_evt3
4	EXTI LINE4	20	LPTIM1_OUT
5	EXTI LINE5	21	LPTIM2_OUT
6	EXTI LINE6	22	LPTIM3_OUT
7	EXTI LINE7	23	Reserved
8	EXTI LINE8	24	Reserved
9	EXTI LINE9	25	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

15.4 DMAMUX functional description

15.4.1 DMAMUX block diagram

Figure 44 shows the DMAMUX block diagram.

Figure 44. 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)

15.4.2 DMAMUX signals

Table 101 lists the DMAMUX signals.

Table 101. DMAMUX signals

Signal name	Description
dmamux_hclk	DMAMUX AHB clock
dmamux_req_inx	DMAMUX DMA request line inputs from peripherals
dmamux_trgx	DMAMUX DMA request triggers inputs (to request generator sub-block)
dmamux_req_genx	DMAMUX request generator sub-block channels outputs
dmamux_reqx	DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels)
dmamux_syncx	DMAMUX synchronization inputs (to request multiplexer sub-block)
dmamux_req_outx	DMAMUX requests outputs (to DMA controllers)
dma_secmx	Secure mode of each DMA controller request channel
dma_privx	Privileged mode of each DMA controller request channel
dmamux_evtx	DMAMUX events outputs
dmamux_non_sec_ovr_it	DMAMUX non-secure overrun interrupts
dmamux_sec_ovr_it	DMAMUX secure overrun interrupts
dmamux_illegal_access_it	DMAMUX security illegal access output (to TrustZone interrupt controller)

15.4.3 DMAMUX channels

A DMAMUX channel is a DMAMUX request multiplexer channel that may include, depending on the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to one single channel of DMA controller(s).

Channel configuration procedure

Follow the sequence below to configure both a DMAMUX x channel and the related DMA channel y:

1. Set to secure or non-secure the DMA channel y by a secure write access to the secure control bit of the DMA channel y configuration register, and set to privileged or unprivileged the DMA channel y by a privileged write access to the privileged control bit of the DMA channel y configuration register.
2. Set and configure completely the DMA channel y, except enabling the channel y.
3. Set and configure completely the related DMAMUX y channel.
4. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

15.4.4 DMAMUX secure/non-secure channels

The DMAMUX is a security-aware peripheral compliant with the TrustZone hardware architecture, partitioning all its resources so that they exist in one of the two worlds: the secure world and the normal/non-secure world, at any given time.

The DMAMUX security is controlled by software at channel level. Any DMAMUX channel is in secure or non-secure state, as configured by the secure register bit of the associated channel of the DMA controller(s).

Note: A DMA controller(s) channel must be first configured as secure or non-secure, before the configuration of the connected DMAMUX channel.

Note: A secure software is able to access any DMAMUX register, whatever secure or non-secure. A non-secure software is restricted to access only non-secure DMAMUX register or non-secure register fields.

A secure read/write access is a read/write transaction on AHB slave with the signal HNONSEC = 0 (at the clock cycle of the address sampling). On the contrary, a non-secure read/write access is a read/write transaction on AHB slave with the signal HNONSEC = 1.

When a channel is configured in secure mode, its configuration register fields become secure resources, meaning that:

- A non-secure read access to a (secure register) field is forced to return 0.
- A non-secure write access to a (secure register) field has no impact.

Additionally, an illegal access signal is generated, as a pulse, to the TrustZone interrupt controller, when a non-secure software attempts to access a secure DMAMUX register:

- DMAMUX_CxCR if the request multiplexer channel x is secure.
- DMAMUX_RGxCR if the request generator channel x is secure.

Note: The secure illegal access signal is never asserted on a non-secure access to the global interrupt status and clear registers, even despite all the DMAMUX channels are set as secure.

15.4.5 DMAMUX privileged / unprivileged channels

The DMAMUX is aware of the privileged or unprivileged state of a given DMA connected channel, and manages consequently its DMAMUX requested channel.

Note: A DMA controller(s) channel must be first configured as privileged or unprivileged, before the configuration of the connected DMAMUX channel.

Note: A privileged software is able to access any DMAMUX register, privileged or unprivileged. An unprivileged software is restricted to access only unprivileged DMAMUX register or register fields.

When a privileged software configures a DMA channel x either as privileged, an unprivileged software is not able to access (write is ignored, read returns zero) the related DMAMUX channel registers or register fields.

15.4.6 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 can be assigned to two different channels only if the application ensures that these channels are not requested to be served at the same time. In other words, if two different channels receive a same asserted hardware request at the same time, an unpredictable DMA hardware behavior occurs.

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 46*, 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 45. Synchronization mode of the DMAMUX request line multiplexer channel

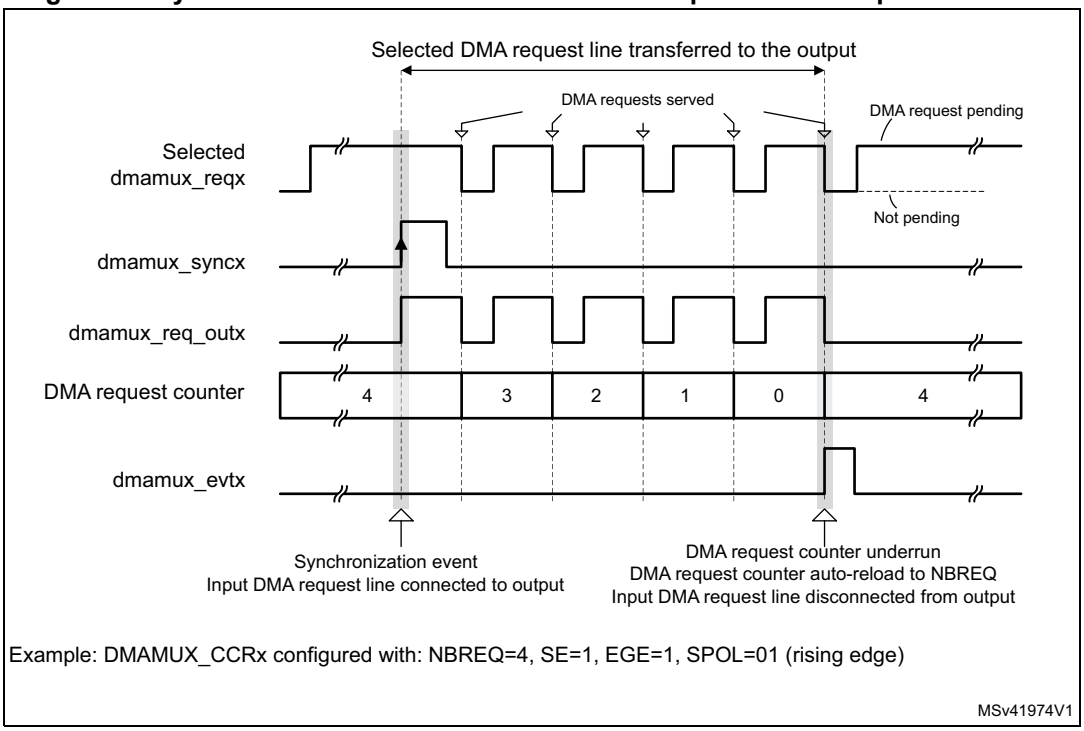
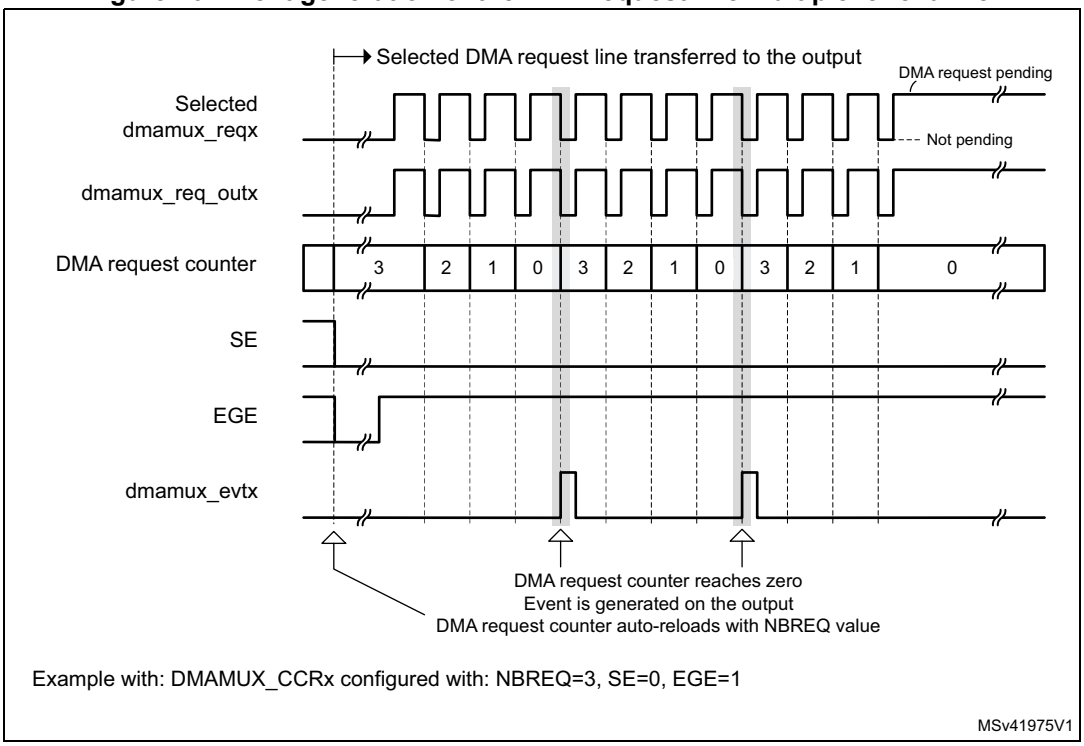


Figure 46. Event generation of the DMA request line multiplexer channel



If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in *Figure 45* and *Figure 46*.

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_CCFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX_CxCR register.

15.4.7 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.

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.

15.5 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. As a consequence, there are mixed secure and non-secure status and clear flag bit fields inside a same global status and clear flag interrupt register, depending on the security of the considered DMAMUX channel.

There are two different secure and non-secure interrupt signals that may be generated, depending on the security of the DMAMUX channel.

Table 102. DMAMUX interrupts

Interrupt signal	Interrupt event	Event flag	Clear bit	Enable bit
dmamux_nonsec_ovr_it	Synchronization event overrun on a non-secure channel x of the DMAMUX request line multiplexer	SOFx	CSOFx	SOIE
	Trigger event overrun on a non-secure channel x of the DMAMUX request generator	OFx	COFx	OIE

Table 102. DMAMUX interrupts (continued)

Interrupt signal	Interrupt event	Event flag	Clear bit	Enable bit
dmamux_sec_ovr_it	Synchronization event overrun on a secure channel x of the DMAMUX request line multiplexer	SOFx	CSOFx	SOIE
	Trigger event overrun on a secure channel x of the DMAMUX request generator	OFx	COFx	OIE

15.6 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.

15.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)

Address offset: 0x000 + 0x04 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register shall be written by a non-secure or secure write, according to the secure mode of the considered DMAMUX request line multiplexer channel x, depending on the secure mode bit of the connected DMA controller channel y. This assumes that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMUX mapping implementation section).

This register shall be accessed by a privileged or unprivileged read/write, according to the privileged mode of the considered DMAMUX request line multiplexer channel x, depending on the privileged control bit of the connected of the connected DMA controller channel y. This assumes that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMUX mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL[1:0]		SE
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	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 100: 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.

15.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR)

Address offset: 0x080

Reset value: 0x0000 0000

This register must be accessed at bit level by a non-secure or secure read, according to the secure mode of the considered DMAMUX request line multiplexer channel x, depending on the secure mode bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMUX mapping implementation section).

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

15.6.3 DMAMUX request line multiplexer interrupt channel clear flag register (DMAMUX_CCFR)

Address offset: 0x084

Reset value: 0x0000 0000

This register must be written at bit level by a non-secure or secure write, according to the secure mode of the considered DMAMUX request line multiplexer channel x, depending on the secure control bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMXUX mapping implementation section).

This register must be written at bit level by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel x,

depending on the privileged control bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMXUX mapping implementation section).

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

15.6.4 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)

Address offset: 0x100 + 0x04 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register shall be written by a non-secure or secure write, according to the secure mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (refer to DMAMUX_CyCR.DMAREQ_ID[7:0] and to the DMAMXUX mapping implementation section).

This register shall be written by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (refer to DMAMUX_CyCR.DMAREQ_ID[7:0] and to the DMAMXUX mapping implementation section).

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

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

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ +1.

Note: This field must be written only when GE bit is disabled.

Bits 18:17 **GPOL[1:0]**: DMA request generator trigger polarity
Defines the edge polarity of the selected trigger input
00: No event, i.e. no trigger detection nor generation.
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 16 **GE**: DMA request generator channel x enable
0: DMA request generator channel x disabled
1: DMA request generator channel x enabled

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

Bit 8 **OIE**: Trigger overrun interrupt enable
0: Interrupt on a trigger overrun event occurrence is disabled
1: Interrupt on a trigger overrun event occurrence is enabled

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **SIG_ID[4:0]**: Signal identification
Selects the DMA request trigger input used for the channel x of the DMA request generator

15.6.5 DMAMUX request generator interrupt status register (DMAMUX_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

This register shall be accessed at bit level by a non-secure or secure read, according to the secure mode of the considered DMAMUX request line multiplexer channel x, depending on the secure mode bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMXUX mapping implementation section).

This register shall be accessed at bit level by an unprivileged or privileged read, according to the privileged mode of the considered DMAMUX request line multiplexer channel x, depending on the privileged control bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMXUX mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF3	OF2	OF1	OF0
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **OF[3:0]**: Trigger overrun event flag
The flag is set when a new trigger event occurs on DMA request generator channel x, before the request counter underrun (the internal request counter programmed via the GNBREQ field of the DMAMUX_RGxCR register).
The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX_RGCFR register.

15.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)

Address offset: 0x144

Reset value: 0x0000 0000

This register shall be written at bit level by a non-secure or secure write, according to the secure mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (refer to DMAMUX_CyCR.DMAREQ_ID[7:0] and to the DMAMXUX mapping implementation section).

This register shall be written at bit level by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (refer to DMAMUX_CyCR.DMAREQ_ID[7:0] and to the DMAMXUX mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF3	COF2	COF1	COF0
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **COF[3:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX_RGSR register.

15.6.7 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 103. 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 103. 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 2.3 on page 87* for the register boundary addresses.

16 Nested vectored interrupt controller (NVIC)

16.1 NVIC main features

- 109 maskable interrupt channels (not including the 16 Cortex®-M33 with FPU interrupt lines)
- 8 programmable priority levels (3 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.

The NVIC registers are banked across secure and non-secure states.

All interrupts including the core exceptions are managed by the NVIC.

16.2 SysTick calibration value register

The Cortex®-M33 with TrustZone mainline security extension embeds two SysTick timers.

When TrustZone is activated, two SysTick timer are available:

- SysTick, secure instance.
- SysTick, non-secure instance.

When TrustZone is disabeld, only one SysTick timer is available.

The SysTick timer calibration value (STCALIB) is 0x3E8. It gives a reference time base of 1 ms based on a SysTick clock frequency of 1 MHz. In order to match the 1 ms time base for an application Running at a given frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- SysTick clock source is CPU clock HCLK: reload value = (HCLK x STCALIB)-1
- SysTick clock source is external clock (HCLK/8): reload value = ((HCLK/8) x STCALIB)-1

The HCLK refers to AHB frequency value in MHz.

Example: SysTick clock source is CPU clock HCLK of 100 MHz, to match a time base of 1 ms:

- SysTick reload value = (100 x STCALIB) -1 = 0x1869F

16.3 Interrupt and exception vectors

The grey rows in *Table 104* describe the vectors without specific position.

Table 104. STM32L552xx and STM32L562xx vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-4	Fixed	Reset	Reset	0x0000 0004
-	-2	Fixed	NMI	Non maskable interrupt. SRAM parity err + FLASH ECC err + HSE CSS	0x0000 0008
-	-3 or -1	Fixed	Secure HardFault	Secure hard fault	0x0000 000C
-	-1	Fixed	Non-secure HardFault	Non-secure hard fault. 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
-	3	Settable	SecureFault	Secure fault	0x0000 001C
-	-	-	-	Reserved	0x0000 0020 - 0x0000 0028
-	4	-	SVC	System service call via SWI instruction	0x0000 002C
-	5	-	Debug Monitor	Debug monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	6	Settable	PendSV	Pendable request for system service	0x0000 0038
-	7	Settable	SysTick	System tick timer	0x0000 003C
0	8	Settable	WWDG	Window watchdog interrupt	0x0000 0040
1	9	Settable	PDV_PVM	PVD/PVM1/PVM2/PVM3/PVM4 through EXTI lines 16/35/36/37/38 interrupts	0x0000 0044
2	10	Settable	RTC	RTC global interrupts (EXTI line 17)	0x0000 0048
3	11	Settable	RTC_S	RTC secure global interrupts (EXTI line 18)	0x0000 004C
4	12	Settable	TAMP	Tamper global interrupt (EXTI line 19)	0x0000 0050
5	13	Settable	TAMP_S	Tamper secure global interrupt (EXTI line 20)	0x0000 0054
6	14	Settable	FLASH	Flash memory global interrupt	0x0000 0058
7	15	Settable	FLASH_S	Flash memory secure global interrupt	0x0000 005C
8	16	Settable	GTZC	TZIC secure global interrupt	0x0000 0060
9	17	Settable	RCC	RCC global interrupt	0x0000 0064
10	18	Settable	RCC_S	RCC secure global interrupt	0x0000 0068

Table 104. STM32L552xx and STM32L562xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
11	19	Settable	EXTI0	EXTI Line0 interrupt	0x0000 006C
12	20	Settable	EXTI1	EXTI Line1 interrupt	0x0000 0070
13	21	Settable	EXTI2	EXTI Line2 interrupt	0x0000 0074
14	22	Settable	EXTI3	EXTI Line3 interrupt	0x0000 0078
15	23	Settable	EXTI4	EXTI Line4 interrupt	0x0000 007C
16	24	Settable	EXTI5	EXTI Line5 interrupt	0x0000 0080
17	25	Settable	EXTI6	EXTI Line6 interrupt	0x0000 0084
18	26	Settable	EXTI7	EXTI Line7 interrupt	0x0000 0088
19	27	Settable	EXTI8	EXTI Line8 interrupt	0x0000 008C
20	28	Settable	EXTI9	EXTI Line9 interrupt	0x0000 0090
21	29	Settable	EXTI10	EXTI Line10 interrupt	0x0000 0094
22	30	Settable	EXTI11	EXTI Line11 interrupt	0x0000 0098
23	31	Settable	EXTI12	EXTI Line12 interrupt	0x0000 009C
24	32	Settable	EXTI13	EXTI Line13 interrupt	0x0000 00E4
25	33	Settable	EXTI14	EXTI Line14 interrupt	0x0000 00A0
26	34	Settable	EXTI15	EXTI Line15 interrupt	0x0000 00A4
27	35	Settable	DMAMUX1_OVR	DMAMUX1 overRun interrupt	0x0000 00A8
28	36	Settable	DMAMUX1_OVR_S	DMAMUX1 secure overRun interrupt	0x0000 00AC
29	37	Settable	DMA1_CH1	DMA1 channel 1 interrupt	0x0000 00B0
30	38	Settable	DMA1_CH2	DMA1 channel 2 interrupt	0x0000 00B4
31	39	Settable	DMA1_CH3	DMA1 channel 3 interrupt	0x0000 00B8
32	40	Settable	DMA1_CH4	DMA1 channel 4 interrupt	0x0000 00C0
33	41	Settable	DMA1_CH5	DMA1 channel 5 interrupt	0x0000 00C4
34	42	Settable	DMA1_CH6	DMA1 channel 6 interrupt	0x0000 00C8
35	43	Settable	DMA1_CH7	DMA1 channel 7 interrupt	0x0000 00CC
36	44	Settable	DMA1_CH8	DMA1 channel 8 interrupt	0x0000 00D0
37	45	Settable	ADC1_2	ADC1_2 global interrupt	0x0000 00D4
38	46	Settable	DAC	DAC global interrupt	0x0000 00D8
39	47	Settable	FDCAN1_IT0	FDCAN1 Interrupt 0	0x0000 00DC
40	48	Settable	FDCAN1_IT1	FDCAN1 Interrupt 1	0x0000 00E0
41	49	Settable	TIM1_BRK	TIM1 break	0x0000 00E4
42	50	Settable	TIM1_UP	TIM1 update	0x0000 00E8
43	51	Settable	TIM1_TRG_COM	TIM1 trigger and commutation	0x0000 00EC

Table 104. STM32L552xx and STM32L562xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
44	52	Settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00F0
45	53	Settable	TIM2	TIM2 global interrupt	0x0000 00F4
46	54	Settable	TIM2	TIM3 global interrupt	0x0000 00F8
47	55	Settable	TIM2	TIM4 global interrupt	0x0000 00FC
48	56	Settable	TIM2	TIM5 global interrupt	0x0000 0100
49	57	Settable	TIM2	TIM6 global interrupt	0x0000 0104
50	58	Settable	TIM2	TIM7 global interrupt	0x0000 0108
51	59	Settable	TIM8_BRK	TIM8 break interrupt	0x0000 010C
52	60	Settable	TIM8_UP	TIM8 update interrupt	0x0000 0110
53	61	Settable	TIM8_TRG_COM	TIM8 trigger and commutation interrupt	0x0000 0114
54	62	Settable	TIM8_CC	TIM8 capture compare interrupt	0x0000 0118
55	63	Settable	I2C1_EV	I2C1 event interrupt	0x0000 011C
56	64	Settable	I2C1_ER	I2C1 error interrupt	0x0000 0120
57	65	Settable	I2C2_EV	I2C2 event interrupt	0x0000 0124
58	66	Settable	I2C2_ER	I2C2 error interrupt	0x0000 0128
59	67	Settable	SPI1	SPI1 global interrupt	0x0000 012C
60	68	Settable	SPI2	SPI2 global interrupt	0x0000 0130
61	69	Settable	USART1	USART1 global interrupt	0x0000 0134
62	70	Settable	USART2	USART2 global interrupt	0x0000 0138
63	71	Settable	USART3	USART3 global interrupt	0x0000 013C
64	72	Settable	UART4	UART4 global interrupt	0x0000 0140
65	73	Settable	UART5	UART5 global interrupt	0x0000 0144
66	74	Settable	LPUART1	LPUART1 global interrupt	0x0000 0148
67	75	Settable	LPTIM1	LPTIM1 global interrupt	0x0000 014C
68	76	Settable	LPTIM2	LPTIM2 global interrupt	0x0000 0150
69	77	Settable	TIM15	TIM15 global interrupt	0x0000 0154
70	78	Settable	TIM16	TIM16 global interrupt	0x0000 0158
71	79	Settable	TIM17	TIM16 global interrupt	0x0000 015C
72	80	Settable	COMP	COMP1/COMP2 through EXTI lines 21/22 interrupts	0x0000 0160
73	81	Settable	USB_FS	USB FS global interrupt	0x0000 0164
74	82	Settable	CRS	Clock recovery system global interrupt	0x0000 0168
75	83	Settable	FMC	FMC global interrupt	0x0000 016C

Table 104. STM32L552xx and STM32L562xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
76	84	Settable	OCTOSPI1	OCTOSPI1 global interrupt	0x0000 0170
77	85	Settable	-	Reserved	0x0000 0174
78	86	Settable	SDMMC1	SDMMC1 global interrupt	0x0000 0178
79	87	Settable	-	Reserved	0x0000 017C
80	88	Settable	DMA2_CH1	DMA2 channel 1 interrupt	0x0000 0180
81	89	Settable	DMA2_CH2	DMA2 channel 2 interrupt	0x0000 0184
82	90	Settable	DMA2_CH3	DMA2 channel 3 interrupt	0x0000_0188
83	91	Settable	DMA2_CH4	DMA2 channel 4 interrupt	0x0000 0174
84	92	Settable	DMA2_CH5	DMA2 channel 5 interrupt	0x0000 0178
85	93	Settable	DMA2_CH6	DMA2 channel 6 interrupt	0x0000 017C
86	94	Settable	DMA2_CH7	DMA2 channel 7 interrupt	0x0000 0180
87	95	Settable	DMA2_CH8	DMA2 channel 8 interrupt	0x0000 0184
88	96	Settable	I2C3_EV	I2C3 event interrupt	0x0000 0188
89	97	Settable	I2C3_ER	I2C3 error interrupt	0x0000 018C
90	98	Settable	SAI1	SAI1 global interrupt	0x0000 0190
91	99	Settable	SAI2	SAI2 global interrupt	0x0000 0194
92	100	Settable	TSC	TSC global interrupt	0x0000 0198
93	101	Settable	AES	AES global interrupt	0x0000 019C
94	102	Settable	RNG	RNG global interrupt	0x0000 01A0
95	103	Settable	FPU	Floating point interrupt	0x0000 01A4
96	104	Settable	HASH	HASH interrupt	0x0000 01A8
97	105	Settable	PKA	PKA global interrupt	0x0000 01AC
98	106	Settable	LPTIM3	LPTIM3 global interrupt	0x0000 01A0
99	107	Settable	SPI3	SPI3 global interrupt	0x0000 01A4
100	108	Settable	I2C4_ER	I2C4 error interrupt	0x0000 01A8
101	109	Settable	I2C4_EV	I2C4 event interrupt	0x0000 01AC
102	110	Settable	DFSDM1_FLT0	DFSDM1 filter 0 global interrupt	0x0000 01B0
103	111	Settable	DFSDM1_FLT1	DFSDM1 filter 0 global interrupt	0x0000 01B4
104	112	Settable	DFSDM1_FLT2	DFSDM1 filter 0 global interrupt	0x0000 01B8
105	113	Settable	DFSDM1_FLT3	DFSDM1 filter 0 global interrupt	0x0000 01BC
106	114	Settable	UCPD1	UCPD1 global interrupt	0x0000 01C0
107	115	Settable	ICACHE	Instruction cache global interrupt	0x0000 01C4
108	116	Settable	OTFDEC1	OTFDEC1 secure global interrupt	0x0000 01C8

17 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wakeup through configurable and direct event inputs. It provides wakeup requests to the power control, and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wakeup requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can be used also in Run modes.

The EXTI also includes the EXTI mux IOport selection.

17.1 EXTI main features

The EXTI main features are the following:

- 43 input events supported.
- All event inputs allow the possibility to wake up the system.
- Events which do not have an associated wakeup flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI.
- Events can be used to generate a CPU wakeup event.

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse).
  The configurable events have the following features:
  - Selectable active trigger edge
  - Interrupt pending status register bits independent for the rising and falling edge
  - Individual interrupt and event generation mask, used for conditioning the CPU wakeup, interrupt and event generation
  - SW trigger possibility
- Direct events (interrupt and wakeup sources from peripherals having an associated flag which requires to be cleared in the peripheral).
  The direct events have the following features:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI. (The interrupt pending status flag is provided by the peripheral generating the event.)
  - Individual interrupt and event generation mask, used for conditioning the CPU wakeup and event generation
  - No SW trigger possibility
- Secure events
  - The access to control and configuration bits of secure input events can be made secure and or privilege.
- EXTI IO port selection.

17.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block, and EXTI mux as shown in *Figure 47*.

The register block contains all the EXTI registers.

The event input trigger block provides event input edge trigger logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, and their masking.

The EXTI mux provides the IO port selection on to the EXTI event signal.

Figure 47. EXTI block diagram

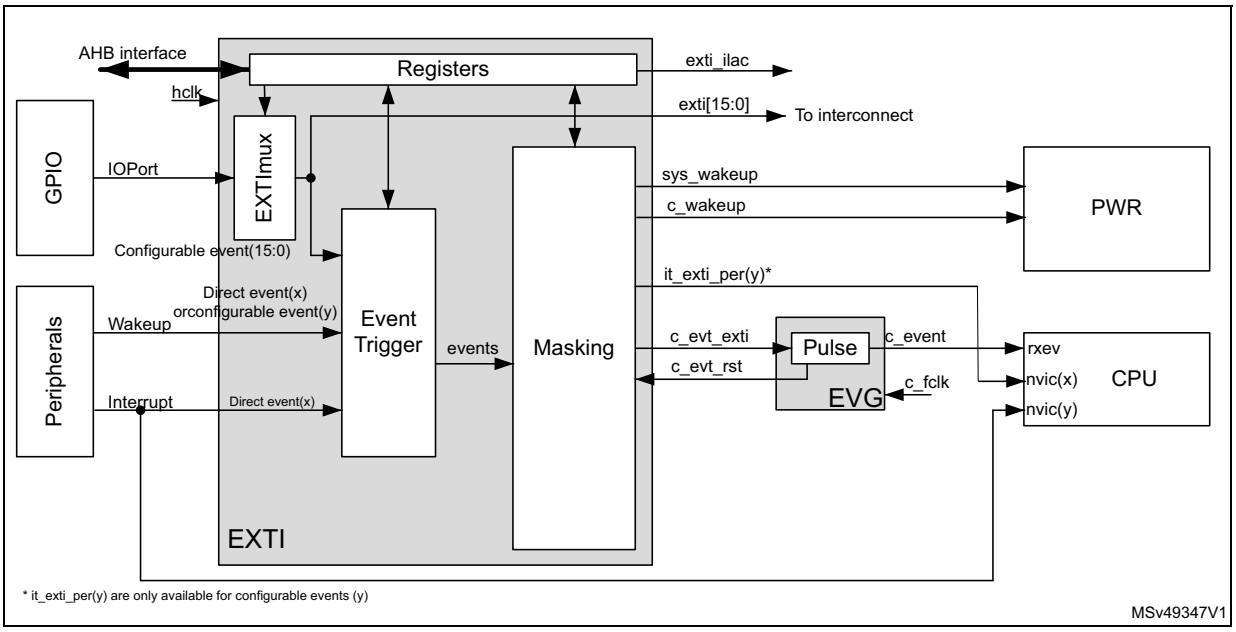


Table 105. EXTI pin overview

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to enable security, the AHB interface supports secure accesses
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wakeup events from peripherals which do not have an associated interrupt and flag in the peripheral
exti_ilac	O	Illegal access event
Direct event(x)	I	Synchronous and asynchronous wakeup events from peripherals which have an associated interrupt and flag in the peripheral
IOPort(n)	I	GPIOs block IO ports[15:0]
exti[15:0]	O	EXTI GPIO output port to trigger other IPs
it_exti_per (y)	O	Interrupts to the CPU associated with Configurable event (y)
c_evt_exti	O	High level sensitive event output for CPU synchronous to hclk

Table 105. EXTI pin overview (continued)

Pin name	I/O	Description
c_evt_rst	I	Asynchronous reset input to clear c_evt_exti
sys_wakeup	O	Asynchronous system wakeup request to PWR for ck_sys and hclk
c_wakeup	O	Wakeup request to PWR for CPU, synchronous to hclk

Table 106. EVG pin overview

Pin name	I/O	Description
c_fclk	I	CPU free Running clock
c_evt_in	I	High level sensitive events input from EXTI, asynchronous to CPU clock
c_event	O	Event pulse, synchronous to CPU clock
c_evt_rst	O	Event reset signal, synchronous to CPU clock

17.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate wakeup or interrupt events when the system is in Stop mode are connected to the EXTI.

- Peripheral wakeup signals which generate a pulse or which do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events the EXTI provides a status pending bit which requires to be cleared. It is the EXTI interrupt associated with the status bit that will interrupt the CPU.
- Peripheral interrupt and wakeup signals that have a status bit in the peripheral which requires to be cleared in the peripheral, are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wakeup is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.
- All GPIO ports input to the EXTI multiplexer, allowing the selection of a port pin to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC of the CPU.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wakeup signals are connected to the PWR block, and are used to wake up the system and CPU sub-system bus clocks.

17.2.2 EXTI interrupt/event mapping

The EXTI lines are connected as shown in *Table 107: EXTI line connections*.

Table 107. EXTI line connections

EXTI line	Line source	Line type
0-15	GPIO	Configurable
16	PVD output	Configurable
17	RTC	Direct
18	RTC secure	Direct
19	TAMP	Direct

Table 107. EXTI line connections (continued)

EXTI line	Line source	Line type
20	TAMP secure	Direct
21	COMP1 output	Configurable
22	COMP2 output	Configurable
23	I2C1 wakeup	Direct
24	I2C2 wakeup	Direct
25	I2C3 wakeup	Direct
26	USART1 wakeup	Direct
27	USART2 wakeup	Direct
28	USART3 wakeup	Direct
29	USART4 wakeup	Direct
30	USART5 wakeup	Direct
31	LPUART1 wakeup	Direct
32	LPTIM1	Direct
33	LPTIM2	Direct
34	USB FS wakeup	Direct
35	PVM1 wakeup	Configurable
36	PVM2 wakeup	Configurable
37	PVM3 wakeup	Configurable
38	PVM4 wakeup	Configurable
39	reserved	Direct
40	I2C4 wakeup	Direct
41	UCPD1 wakeup	Direct
42	LPTIM3 wakeup	Direct

17.3 EXTI functional description

Depending on the EXTI event input type and wakeup target(s), different logic implementations are used. The applicable features are controlled from register bits:

- Active trigger edge enable, by rising edge selection
  EXTI rising trigger selection register (EXTI_RTSR1),
  EXTI rising trigger selection register (EXTI_RTSR2),
  and falling edge selection
  EXTI falling trigger selection register (EXTI_FTSR1),
  EXTI falling trigger selection register (EXTI_FTSR2).
- Software trigger, by
  EXTI software interrupt event register (EXTI_SWIER1),
  EXTI software interrupt event register (EXTI_SWIER2).
- Interrupt pending flag, by
  EXTI rising edge pending register (EXTI_RPR1),
  EXTI falling edge pending register (EXTI_FPR1),
  EXTI rising edge pending register (EXTI_RPR2),
  EXTI falling edge pending register (EXTI_FPR2).
- CPU wakeup and interrupt enable, by
  EXTI CPU wakeup with interrupt mask register (EXTI_IMR1),
  EXTI CPU wakeup with interrupt mask register (EXTI_IMR2).
- CPU wakeup and event enable, by
  EXTI CPU wakeup with interrupt mask register (EXTI_IMR1),
  EXTI CPU wakeup with event mask register (EXTI_EMR2).

Table 108. EXTI event input configurations and register control

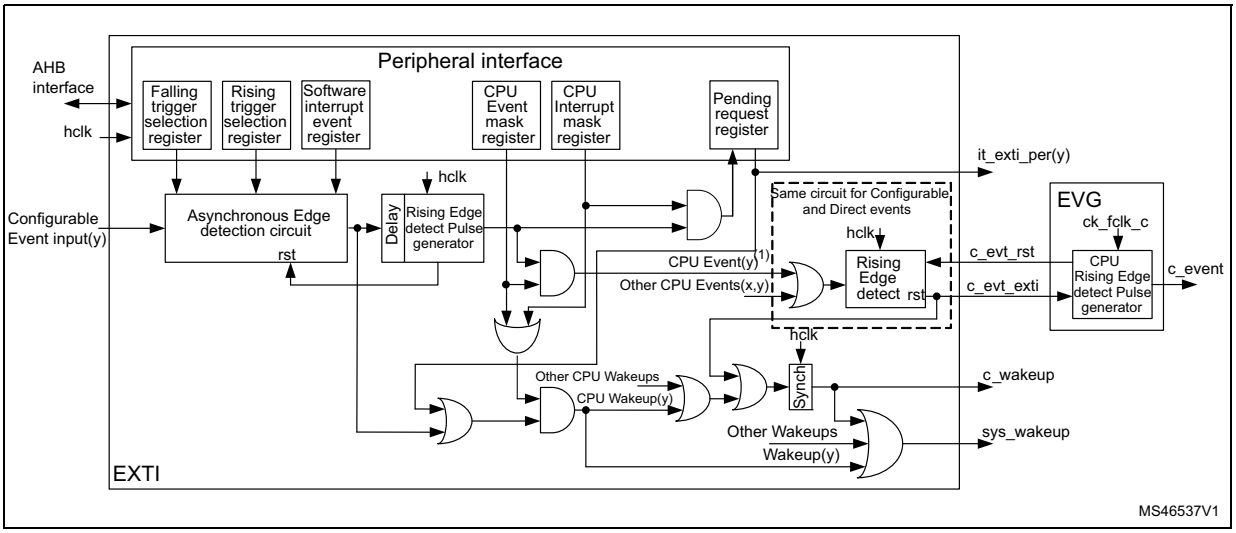
Event input type	Logic implementation	EXTI_RTSR	EXTI_FTSR	EXTI_SWIER	EXTI_R/FPR	EXTI_IMR	EXTI_EMR(1)
Configurable	Configurable event input wakeup logic	x	x	x	x	x	x
Direct	Direct event input wakeup logic	-	-	-	-	x	x

1. Only for input events with configuration “rxev generation” enabled.

17.3.1 EXTI configurable event input wakeup

Figure 48 is a detailed representation of the logic associated with configurable event inputs which wakeup the CPU sub-system bus clocks and generated an EXTI pending flag and interrupt to the CPU and or a CPU wakeup event.

Figure 48. Configurable event trigger logic CPU wakeup



1. Only for the input events that support CPU rxev generation c_event.

The software interrupt event register allows to trigger configurable events by software, writing the corresponding register bit, irrespective of the edge selection setting.

The rising edge and falling edge selection registers allow the enabling and selection of the configurable event active trigger edge or both edges.

The CPU has its dedicated wakeup (interrupt) mask register and a dedicated event mask registers. The enabled event make it possible to generate an event on the CPU. All events for a CPU are ordered together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) is not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers, shared by the CPU. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts need to be acknowledged by software in the EXTI_RPR and/or EXTI_FPR registers.

When a CPU wakeup (interrupt) or CPU event is enabled the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

Note: A detected configurable event interrupt pending request, may be cleared by any CPU with the correct access permission. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

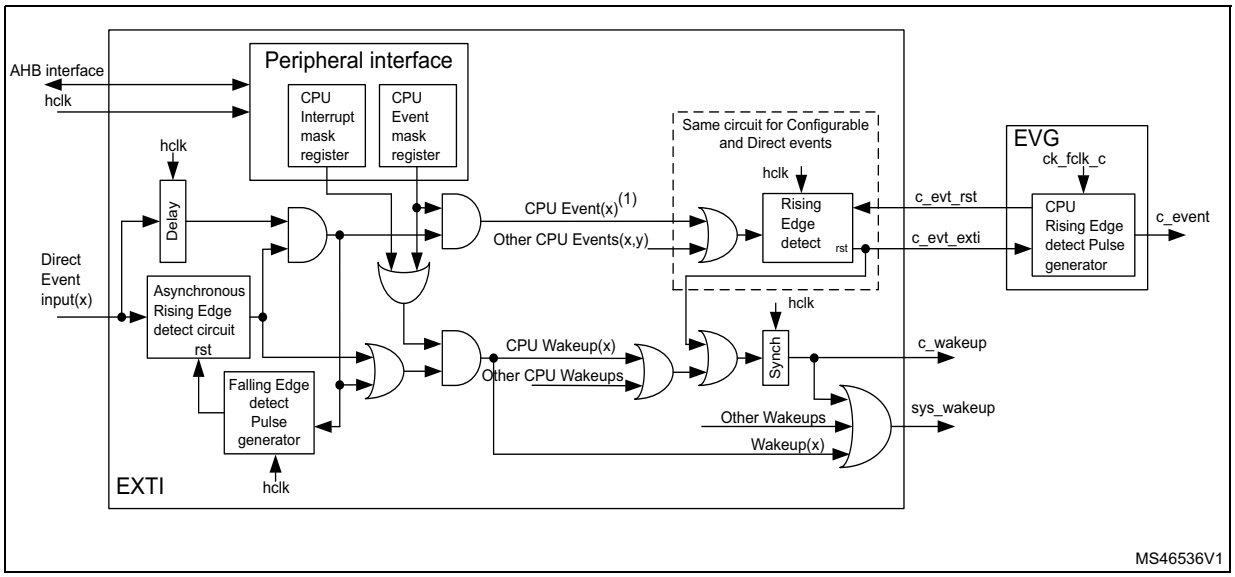
17.3.2 EXTI direct event input wakeup

Figure 49 is a detailed representation of the logic associated with direct event inputs waking up the system.

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU sub-system clocks and may generate a CPU wakeup event. The peripheral synchronous interrupt, associated with the direct wakeup event wakes up the CPU.

The EXTI direct event is able to generate a CPU event. This CPU event wakes up the CPU. The CPU event may occur before the associated peripheral interrupt flag is set.

Figure 49. Direct event trigger logic CPU wakeup

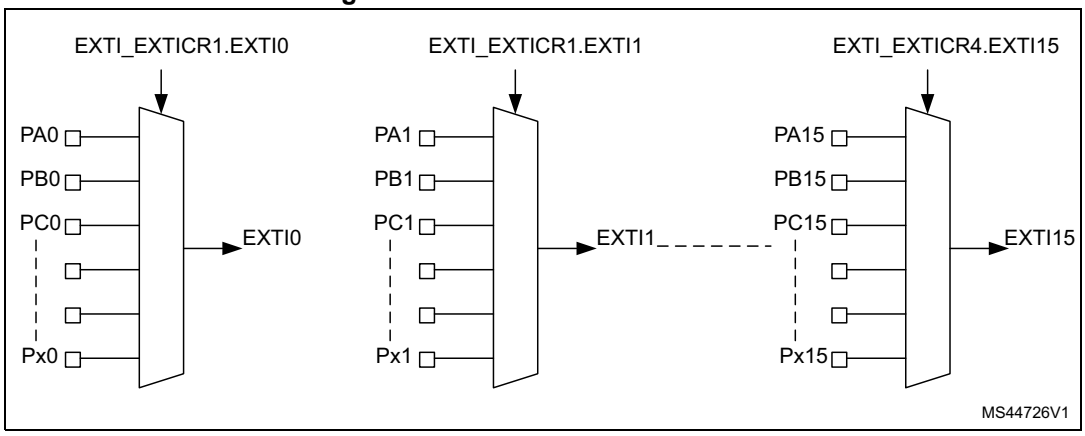


1. Only for the input events that support CPU rxev generation c_event.

17.3.3 EXTI mux selection

The EXTI mux allow the selection of GPIOs as interrupts and wakeup. The GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI mux output, is controlled by registers: *EXTI external interrupt selection register (EXTI_EXTICRn)*.

Figure 50. EXTI mux GPIO selection



The EXTIs mux outputs are available as output signals from the EXTI to trigger other IPs. The EXTI mux outputs are available independent from any masking in EXTI_IMRx and EXTI_EMRx.

17.4 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wakeup event. The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the generation of a CPU wakeup is conditioned by the CPU interrupt mask and CPU event mask.

Table 109. Masking functionality

CPU interrupt enable EXTI_IMR.IMn	CPU event enable EXTI_EMR.EMn	Configurable event inputs EXTI_RPR.RPIFn EXTI_FPR.FPIFn	exti(n) interrupt(1)	CPU event	CPU wakeup
0	0	no	masked	masked	masked
	1	no	masked	yes	yes
1	0	status latched	yes	masked	yes(2)
	1	status latched	yes	yes	yes

1. The single exti(n) interrupt will go to the CPU. If no interrupt is required for CPU, the exti(n) interrupt shall be masked in the CPU NVIC.
2. Only if CPU interrupt is enabled in EXTI_IMR.IMn.

For configurable event inputs, when the enabled edge(s) occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked the corresponding pending bit EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn is/are set and the CPU sub-system is woken up and CPU interrupt signal is activated. The EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn pending bit shall be cleared by software writing it to ‘1’. This action clears the CPU interrupt.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit in the EXTI.

When the associated CPU interrupt is unmasked, the corresponding CPU sub-system is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU event has to be unmasked to generate an event. When the enabled edge(s) occur on the event input a CPU event pulse is generated. There is no event pending bit.

For the configurable event inputs an event request can be generated by software when writing a ‘1’ in the software interrupt/event register EXTI_SWIER, allowing the generation of a rising edge on the event. The rising edge event pending bit is set in EXTI_RPR, irrespective of the setting in EXTI_RTSR.

17.5 EXTI event protection

The EXTI is able to protect event register bits from being modified by non-secure and unprivileged accesses. The protection can individually be activated per input event via the register bits in EXTI_SECCFGR and EXTI_PRIVCFGR. At EXTI level the protection consists in preventing unauthorized write access to:

- Change the settings of the secure and/or privileged configurable events.
- Change the masking of the secure and/or privileged input events.
- Clear pending status of the secure and/or privileged input events.

Table 110. Register protection overview

Register name	Access type	Protection(1)(2)
EXTI_RTSR	RW	Security and privilege can be bit wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR
EXTI_FTSR	RW	
EXTI_SWIER	RW	
EXTI_RPR	RW	
EXTI_FPR	RW	
EXTI_SECCFGR	RW	Always secure, and privilege can be bit wise enabled in EXTI_PRIVCFGR
EXTI_PRIVCFGR	RW	Always privilege, and security can be bit wise enabled in EXTI_SECCFGR
EXTI_EXTICRn	RW	Security and privilege can be bit wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR
EXTI_LOCKR	RW	Always secure.
EXTI_IM	RW	Security and privilege can be bit wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR
EXTI_EMR	RW	

1. Security is enabled with the individual Input event. EXTI_SECCFG registers.
2. Privilege is enabled with the individual Input event EXTI_PRIVCFGRn registers.

17.5.1 EXTI security protection

When security is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a secure access, a non-secure write access is discarded and a read returns 0.

When input events are non-secure, the security is disabled. The associated input event configuration and control bits can be modified and read by a secure access and non-secure access.

The security configuration in registers EXTI_SECCFGR can be globally locked after reset by EXTI_LOCKR.LOCK.

17.5.2 EXTI privilege protection

When privilege is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a privilege access, an unprivileged write access is discarded and a read returns 0.

When input events are unprivileged, the privilege is disabled. The associated input event configuration and control bits can be modified and read by a privilege access and unprivileged access.

The privilege configuration in registers EXTI_PRIVCFGRn can be globally locked after reset by EXTI_LOCKR.LOCK.

17.6 EXTI registers

The EXTI register map is divided in the following sections:

Table 111. EXTI register map sections

Address offset	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [63:32] configuration
0x060 - 0x06C	EXTI IOport mux selection
0x070	EXTI protection lock configuration
0x080 - 0x0BC	CPU input event configuration

All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

17.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	Res.	Res.	Res.	Res.	RT16
									rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 22:21 **RT[22:21]:** Rising trigger event configuration bit of configurable event input x[(1)] (where x = 18 to 22)

When SECx is disabled, RTx can be accessed with non-secure and secure access.

When SECx is enabled, RTx can only be accessed with secure access. Non-secure write to this bit x is discarded, non-secure read returns 0.

When EXTI_PRIVCFGRn.PRIVx is disabled, RTx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGRn.PRIVx is enabled, RTx can only be accessed with privilege access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.

0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

Bits 20:17 Reserved, must be kept at reset value.

Bits 16:0 **RT[16:0]:** Rising trigger event configuration bit of configurable event input x[(1)] (where x = 0 to 16)

When EXTI_SECCFGR SECx is disabled, RTx can be accessed with non-secure and secure access.

When EXTI_SECCFGR SECx is enabled, RTx can only be accessed with secure access. Non-secure write to this bit x is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR PRIVx is disabled, RTx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR PRIVx is enabled, RTx can only be accessed with privilege access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.

0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

17.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	Res.	Res.	Res.	Res.	FT16
									rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 22:21 **FT[22:21]:** Falling trigger event configuration bit of configurable event input x [1] (where x = 18 to 22).

When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access. Non-secure write to this FTx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privilege access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.

0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bit 20:17 Reserved, must be kept at reset value.

Bits 16:0 **FT[16:0]:** Falling trigger event configuration bit of configurable event input x [1] (where x = 0 to 16).

When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access. Non-secure write to this FTx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privilege access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.

0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

17.6.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI22	SWI21	Res.	Res.	Res.	Res.	SWI16
									rw	rw					rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 22:21 **SWI[22:21]:** Software interrupt on event x (where x = 18 to 22)

When EXTI_SECFGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access. Non-secure write to this SWI x is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privilege access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit triggers a rising edge event on event x. This bit is auto cleared by HW.

Bit 20:17 Reserved, must be kept at reset value.

Bits 16:0 **SWI[16:0]:** Software interrupt on event x (where x = 18 to 22)

When EXTI_SECFGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access. Non-secure write to this SWI x is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privilege access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit triggers a rising edge event on event x. This bit is auto cleared by HW.

17.6.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF22	RPIF21	Res.	Res.	Res.	Res.	RPIF16
									rc_w1	rc_w1					rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
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 31:23 Reserved, must be kept at reset value.

Bits 22:21 **RPIF[22:21]:** configurable event inputs x rising edge pending bit (where x = 18 to 22).

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access. Non-secure write to this RPIFx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privilege access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.

0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 20:17 Reserved, must be kept at reset value.

Bits 16:0 **RPIF[16:0]:** configurable event inputs x rising edge pending bit (where x = 0 to 16).

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access. Non-secure write to this RPIFx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privilege access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.

0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

17.6.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF22	FPIF21	Res.	Res.	Res.	Res.	FPIF16
									rc_w1	rc_w1					rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
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 31:23 Reserved, must be kept at reset value.

Bits 22:21 **FPIF[22:21]:** configurable event inputs x falling edge pending bit (where x = 22 to 18)

When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privilege access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.

0: No falling edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 20:17 Reserved, must be kept at reset value.

Bits 16:0 **FPIF[16:0]:** configurable event inputs x falling edge pending bit (where x = 0 to 16)

When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privilege access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.

0: No falling edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

17.6.6 EXTI security configuration register (EXTI_SECCFGR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register provides write access security, a non-secure write access is ignored and causes the generation of an illegal access event. A non-secure read returns the register data.

Contains only register bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
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
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC[31:0]:** Security enable on event input x (where x = 0 to 31)

When EXTI_PRIVCFGRn.PRIVx is disabled, SECx can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGRn.PRIVx is enabled, SECx can only be written with privilege access. Unprivileged write to this SECx is discarded.

0: Event security disabled (non-secure)
1: Event security enabled (secure)

17.6.7 EXTI privilege configuration register (EXTI_PRIVCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
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
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV[31:0]:** Security enable on event input x (where x = 0 to 31)

When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and non-secure access.

When EXTI_SECCFGR.SECx is enabled, PRIVx can only be written with secure access. Non-secure write to this PRIVx is discarded.

0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

17.6.8 EXTI rising trigger selection register (EXTI_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT38	RT37	RT36	RT35	Res.	Res.	Res.
									rw	rw	rw	rw			

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:3 **RT[38:35]:** Rising trigger event configuration bit of configurable event input x(1) (where x = 35 to 38).

When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, RTx can only be accessed with secure access. Non-secure write to this RTx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, RTx can only be accessed with privilege access. Unprivileged write to this RTx is discarded, unprivileged read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bits 2:0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

17.6.9 EXTI falling trigger selection register (EXTI_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT38	FT37	FT36	FT35	Res.	Res.	Res.
									rw	rw	rw	rw			

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:3 **FT[38:35]:** Falling trigger event configuration bit of configurable event input x(1) (where x = 35 to 38)

When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access. Non-secure write to this FTx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privilege access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bits 2:0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

17.6.10 EXTI software interrupt event register (EXTI_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI38	SWI37	SWI36	SWI35	Res.	Res.	Res.
									rw	rw	rw	rw			

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:3 **SWI[38:35]:** Software interrupt on event x (where x = 35 to 38)

When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access. Non-secure write to this SWIx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privilege access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit triggers a rising edge event on event x+32. This bit is auto cleared by HW.

Bits 2:0 Reserved, must be kept at reset value.

17.6.11 EXTI rising edge pending register (EXTI_RPR2)

Address offset: 0x02C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF38	RPIF37	RPIF36	RPIF35	Res.	Res.	Res.
									rc_w1	rc_w1	rc_w1	rc_w1			

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:3 **RPIF[38:35]:** configurable event inputs x rising edge pending bit (where x = 35 to 38).
When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access. Non-secure write to this RPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIF can be accessed with unprivileged and privilege access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privilege access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bits 2:0 Reserved, must be kept at reset value.

17.6.12 EXTI falling edge pending register (EXTI_FPR2)

Address offset: 0x030

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF38	FPIF37	FPIF36	FPIF35	Res.	Res.	Res.
									rc_w1	rc_w1	rc_w1	rc_w1			

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:3 **FPIF[38:35]:** configurable event inputs x pending bit (where x = 35 to 38).

When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privilege access.

When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privilege access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.

0: No fallling edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bits 2:0 Reserved, must be kept at reset value.

17.6.13 EXTI security enable register (EXTI_SECCFGR2)

Address offset: 0x034

Reset value: 0x0000 0000

This register provides write access security, a non-secure write access is ignored and causes the generation of an illegal access event. A non-secure read returns the register data.

Contains only register bits for security capable Input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SEC42	SEC41	SEC40	SEC39	SEC38	SEC37	SEC36	SEC35	SEC34	SEC33	SEC32
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **SEC[42:32]:** Security enable on event input x (where x = 32 to 42)

When EXTI_PRIVCFGRn.PRIVx is disabled, SECx can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGRn.PRIVx is enabled, SECx can only be written with privilege access. Unprivileged write to this SECx is discarded

0: Event security disabled (non-secure)
1: Event security enabled (secure)

17.6.14 EXTI privilege enable register (EXTI_PRIVCFGR2)

Address offset: 0x038

Reset value: 0x0000 0000

This register provides privileged write access protection, an unprivileged write access is discarded and causes the generation of an illegal access event. An unprivileged read returns the register data.

Contains only register bits for security capable Input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	PRIV42	PRIV41	PRIV40	PRIV39	PRIV38	PRIV37	PRIV36	PRIV35	PRIV34	PRIV33	PRIV32
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **PRIV[42:32]:** Privilege enable on event input x (where x = 32 to 42)

When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and non-secure access.

When EXTI_SECCFGR.SECx is enabled, PRIVx can only be accessed with secure access. Non-secure write to this PRIVx is discarded.

0: Event privilege disabled (unprivileged)
1: Event security enabled (privileged)

17.6.15 EXTI external interrupt selection register (EXTI_EXTICRn)

Address offset: 0x060 (EXTI_EXTICR1) EXTI mux 0, 1, 2, 3 (m =0)
Address offset: 0x064 (EXTI_EXTICR2) EXTI mux 4, 5, 6, 7 (m = 4)
Address offset: 0x068 (EXTI_EXTICR3) EXTI mux 8, 9, 10, 11 (m = 8)
Address offset: 0x06C (EXTI_EXTICR4) EXTI mux 12, 13, 14, 15 (m = 12)

Reset value: 0x0000 0000

EXTIm fields contain only the number of bits in line with the nb_ioport configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTIm+3[7:0]								EXTIm+2[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
EXTIm+1[7:0]								EXTIm[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTIm+3[7:0]:** EXTIm+3 GPIO port selection (where m = 0, 4, 8, or 12 for respectively EXTI_EXTICR[1:4]).

These bits are written by software to select the source input for EXTIm+3 external interrupt.

When EXTI_SECCFGR.SEC(m+3) is disabled, EXTI(m+3) can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SEC(m+3) is enabled, EXTI(m+3) can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIV(m+3) is disabled, EXTI(m+3) can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGR.PRIV(m+3) is enabled, EXTI(m+3) can only be accessed with privilege access. Unprivileged write to this bit is discarded.

0x00: PA[m+3] pin
0x01: PB[m+3] pin
0x02: PC[m+3] pin
0x03: PD[m+3] pin
0x04: PE[m+3] pin
0x05: PF[m+3] pin
0x06: PG[m+3] pin
0x07: PH[m+3] pin
Others reserved

Bits 23:1 **EXTIm+2[7:0]:** EXTIm+2 GPIO port selection (where m = 0, 4, 8, or 12 for respectively EXTI_EXTICR[1:4])(where m = 0, 4, 8, or 12 for respectively EXTI_EXTICR[1:4]).

These bits are written by software to select the source input for EXTIm+2 external interrupt.

When EXTI_SECCFGR.SEC(m+2) is disabled, EXTI(m+2) can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SEC(m+2) is enabled, EXTI(m+2) can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIV(m+2) is disabled, EXTI(m+2) can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGR.PRIV(m+2) is enabled, EXTI(m+2) can only be accessed with privilege access. Unprivileged write to this bit is discarded.

0x00: PA[m+2] pin
0x01: PB[m+2] pin
0x02: PC[m+2] pin
0x03: PD[m+2] pin
0x04: PE[m+2] pin
0x05: PF[m+2] pin
0x06: PG[m+2] pin
0x07: PH[m+2] pin
Others reserved

Bits 15:8 **EXTIm+1[7:0]:** EXTIm+1 GPIO port selection (where m = 0, 4, 8, or 12 for respectively EXTI_EXTICR[1:4]).

These bits are written by software to select the source input for EXTIm+1 external interrupt.

When EXTI_SECCFGR.SEC(m+1) is disabled, EXTI(m+1) can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SEC(m+1) is enabled, EXTI(m+1) can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVm+1 is disabled, EXTI(m+1) can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGR.PRIVm+1 is enabled, EXTI(m+1) can only be accessed with privilege access. Unprivileged write to this bit is discarded.

0x00: PA[m+1] pin
0x01: PB[m+1] pin
0x02: PC[m+1] pin
0x03: PD[m+1] pin
0x04: PE[m+1] pin
0x05: PF[m+1] pin
0x06: PG[m+1] pin
0x07: PH[m+1] pin
Others reserved

Bits 7:0 **EXTIm[7:0]:** EXTIm GPIO port selection (where m = 0, 4, 8, or 12 for respectively EXTI_EXTICR[1:4]).

These bits are written by software to select the source input for EXTIm external interrupt.

When EXTI_SECCFGR.SEC(m) is disabled, EXTI(m) can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SEC(m) is enabled, EXTI(m) can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIV(m) is disabled, EXTI(m) can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGR.PRIV(m) is enabled, EXTI(m) can only be accessed with privilege access. Unprivileged write to this bit is discarded

0x00: PA[m] pin
0x01: PB[m] pin
0x02: PC[m] pin
0x03: PD[m] pin
0x04: PE[m] pin
0x05: PF[m] pin
0x06: PG[m] pin
0x07: PH[m] pin
Others reserved

17.6.16 EXTI lock register (EXTI_LOCKR)

Address offset: 0x070

Reset value: 0x0000 0000

This register provides both write access security, a non-secure write access is ignored and a read access returns zero data, and generate an illegal access event.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LOCK**: Global security and privilege configuration registers EXTI_SECCFGR and EXTI_PRIVCFGR lock.

This register bit is write once after reset.

0: Security and privilege configuration open, can be modified.
1: Security and privilege configuration locked, can no longer be modified.

17.6.17 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0xFF9E 0000

Contains register bits for configurable events and direct events.

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 **IM[31:0]:** CPU wakeup with interrupt mask on event input x [(1)] (where x = 0 to 31).

When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Non-secure write to this bit is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privilege access. Unprivileged write to this bit is discarded.

0: Wakeup with interrupt request from input event x is masked
1: Wakeup with interrupt request from input event x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.

17.6.18 EXTI CPU wakeup with event mask register (EXTI_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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 **EM[31:0]:** CPU wakeup with event generation mask on event input x (where x = 0 to 31).

When EXTI_SECCFGR.SECENx is disabled, EMx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECENx is enabled, EMx can only be accessed with secure access. Non-secure write to this bit x is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privilege access. Unprivileged write to this bit is discarded.

0: Wakeup with event generation from Line x is masked
1: Wakeup with event generation from Line x is unmasked

17.6.19 EXTI CPU wakeup with interrupt mask register (EXTI_IMR2)

Address offset: 0x090

Reset value: 0x0000 0787

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	IM42	IM41	IM40	Res.	IM38	IM37	IM36	IM35	IM34	IM33	IM32
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **IM[42:40]:** CPU wakeup with interrupt mask on event input x [(1)] (where x = 40 to 42).

When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Non-secure write to this bit is discarded, non-secure read returns 0..

When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privilege access. Unprivileged write to this bit is discarded.

0: Wakeup with interrupt request from input event x is masked
1: Wakeup with interrupt request from input event x is unmasked

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **IM[38:32]:** CPU wakeup with interrupt mask on event input x[(1)] (where x = 32 to 38).

When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Non-secure write to this bit is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privilege and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privilege access. Unprivileged write to this bit is discarded.

0: Wakeup with interrupt request from input event x is masked
1: Wakeup with interrupt request from input event x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.

17.6.20 EXTI CPU wakeup with event mask register (EXTI_EMR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	EM42	EM41	EM40	Res.	EM38	EM37	EM36	EM35	EM34	EM33	EM32
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **EM[42:40]:** CPU wakeup with event generation mask on event input(1) (where x = 40 to 42).
When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, EMx can only be accessed with secure access. Non-secure write to this bit x is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privilege and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privilege access. Unprivileged write to this bit is discarded.
0: Wakeup with event generation from line x is masked
1: Wakeup with event generation from line x is unmasked

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **EM[38:32]:** CPU wakeup with event generation mask on event input x(1) (where x = 32 to 38).
When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, EMx can only be accessed with secure access. Non-secure write to this bit x is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privilege and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privilege access. Unprivileged write to this bit is discarded.
0: Wakeup with event generation from line x is masked
1: Wakeup with event generation from line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.

17.6.21 EXTI register map

The following table gives the EXTI register map and the reset values.

Table 112. 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
0x000	**EXTI_RTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT[22:21]		Res.	Res.	Res.	Res.	RT[16:0]																
	Reset value										0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 112. Extended interrupt/event controller 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	**EXTI_FTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT[22:21]		Res.	Res.	Res.	Res.	FT[16:0]																
	Reset value										0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**EXTI_SWIER1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI[22:21]		Res.	Res.	Res.	Res.	SWi[16:0]																
	Reset value										0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**EXTI_RPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF[22:21]		Res.	Res.	Res.	Res.	RPIF[16:0]																
	Reset value										0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**EXTI_FPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF[22:21]		Res.	Res.	Res.	Res.	FPIF[16:0]																
	Reset value										0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**EXTI_SECCFGR1**	SEC[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	**EXTI_PRIVCFG1**	PRIV[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**EXTI_RTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT [38:35]				Res.	Res.	Res.
	Reset value																										0	0	0	0			
0x024	**EXTI_FTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT [38:35]				Res.	Res.	Res.
	Reset value																										0	0	0	0			
0x028	**EXTI_SWIER2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI [38:35]				Res.	Res.	Res.
	Reset value																										0	0	0	0			
0x02C	**EXTI_RPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF [38:35]				Res.	Res.	Res.
	Reset value																										0	0	0	0			
0x030	**EXTI_FPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF [38:35]				Res.	Res.	Res.
	Reset value																										0	0	0	0			
0x034	**EXTI_SECCFG2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC[42:32]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x038	**EXTI_PRIVCFG2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV[42:32]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Table 112. Extended interrupt/event controller 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
0x060	**EXTI_EXTICR1**	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]								EXTI0[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
0x064	**EXTI_EXTICR2**	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]								EXTI4[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
0x068	**EXTI_EXTICR3**	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]								EXTI8[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
0x06C	**EXTI_EXTICR4**	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]								EXTI2[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
0x070	**EXTI_LOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK
	Reset value																																0
0x080	**EXTI_IMR1**	IM[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	**EXTI_EMR1**	EM[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
0x088-0x08C	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.
0x090	**EXTI_IMR2**																						IM[42:32]										
	Reset value																						1	1	1	1	0	0	0	0	1	1	1
0x094	**EXTI_EMR2**																						EM[42:32]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3 on page 87* for the register boundary addresses.

18 Cyclic redundancy check calculation unit (CRC)

18.1 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.

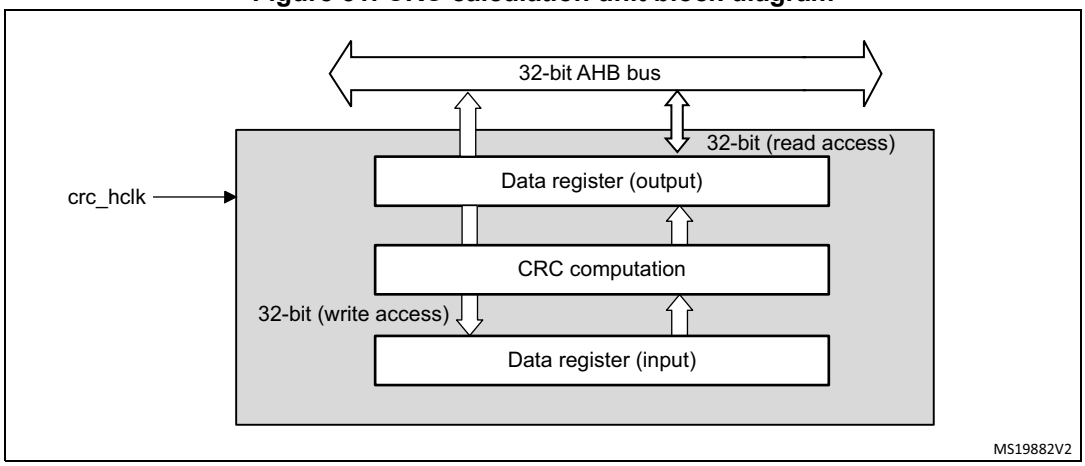
18.2 CRC main features

- 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

18.3 CRC functional description

18.3.1 CRC block diagram

Figure 51. CRC calculation unit block diagram



18.3.2 CRC internal signals

Table 113. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

18.3.3 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.

18.4 CRC registers

18.4.1 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.

18.4.2 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

18.4.3 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

18.4.4 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.

18.4.5 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.

18.4.6 CRC register map

Table 114. 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 2.3 on page 87* 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.

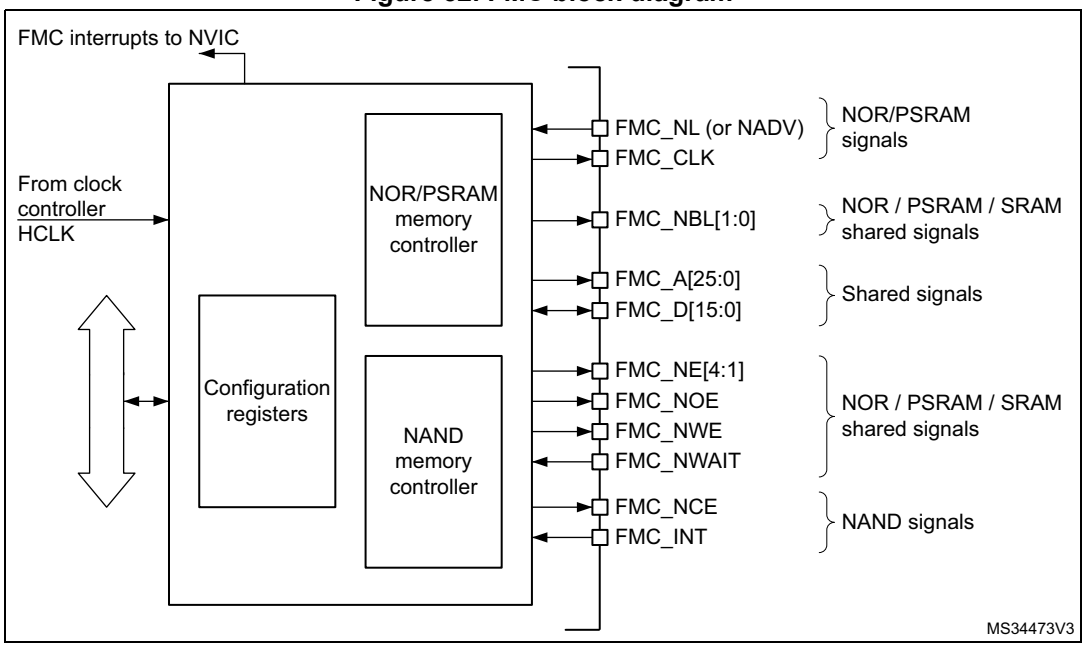
19.3 FMC block diagram

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®-M33 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.

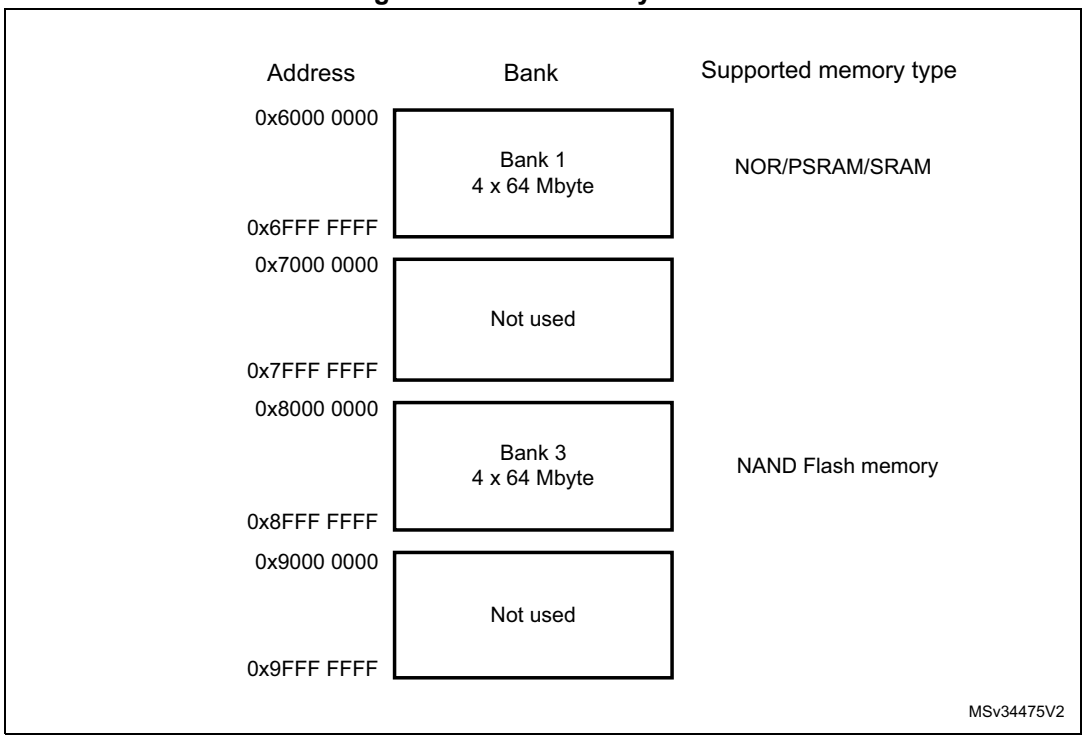
19.5 External device address mapping

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 115*.

Table 115. 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 116. 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 117*.

Table 117. 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 118* 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 118. 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 119*) and support for wait management (for PSRAM and NOR Flash accessed in Burst mode).

Table 119. 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 120, *Table 121* and *Table 122* 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 120. 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 121. 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 122. 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 123. 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 123. 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 124 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 124. 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 124. 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

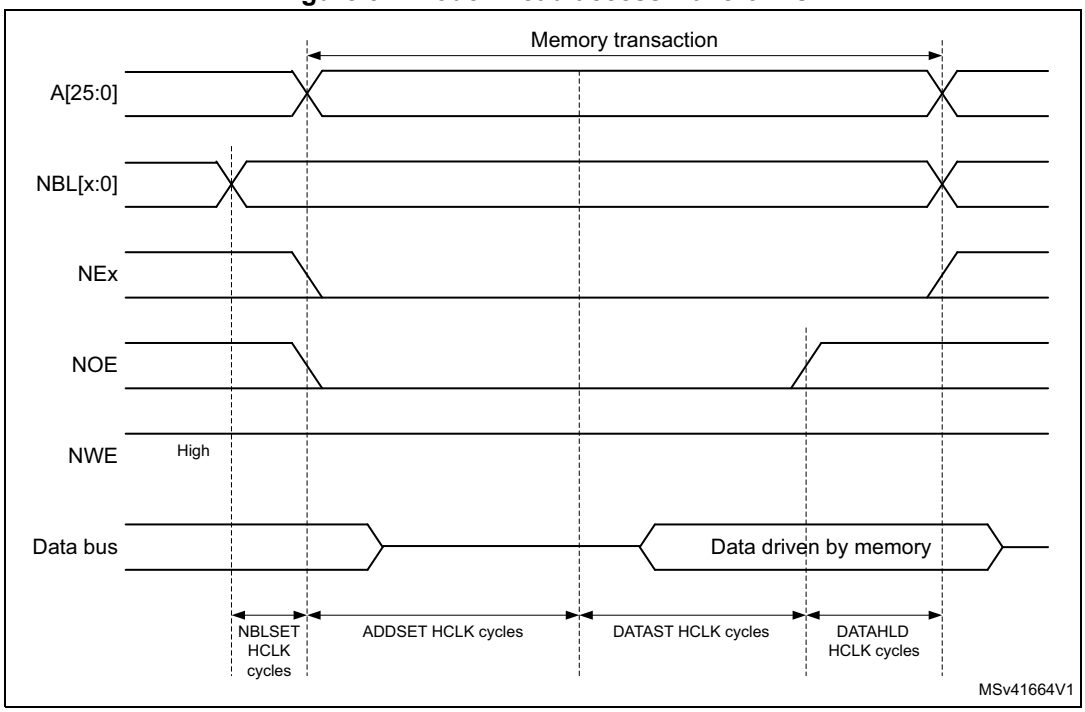
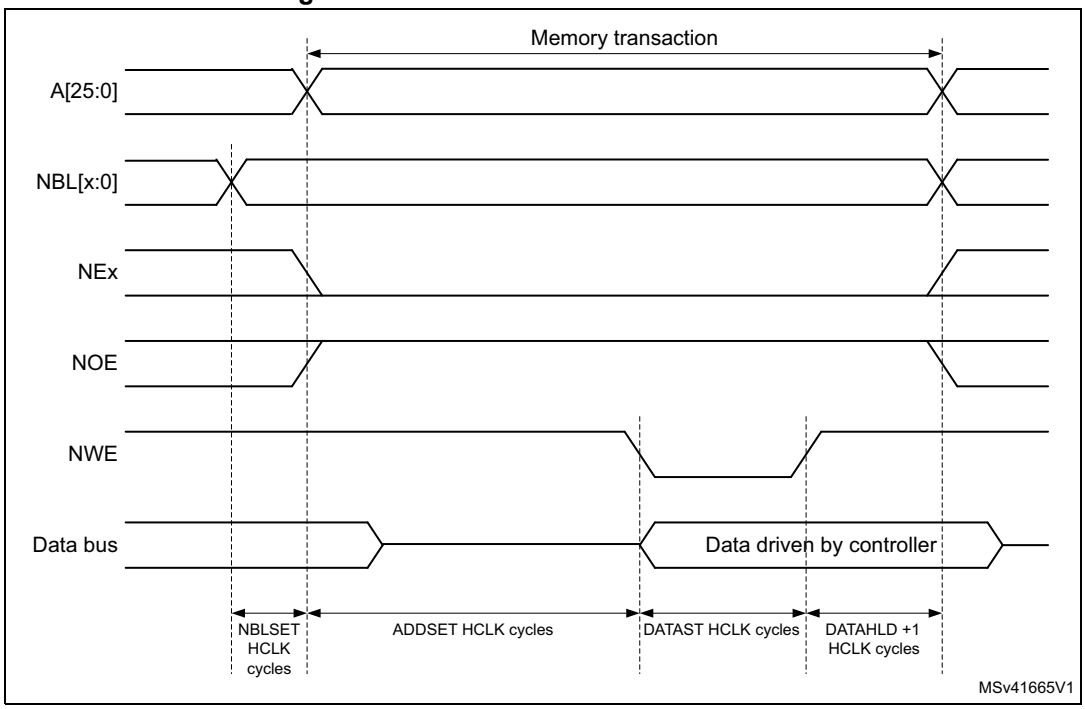


Figure 55. Mode 1 write access waveforms



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 125. 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 125. FMC_BCRx bitfields (mode 1) (continued)

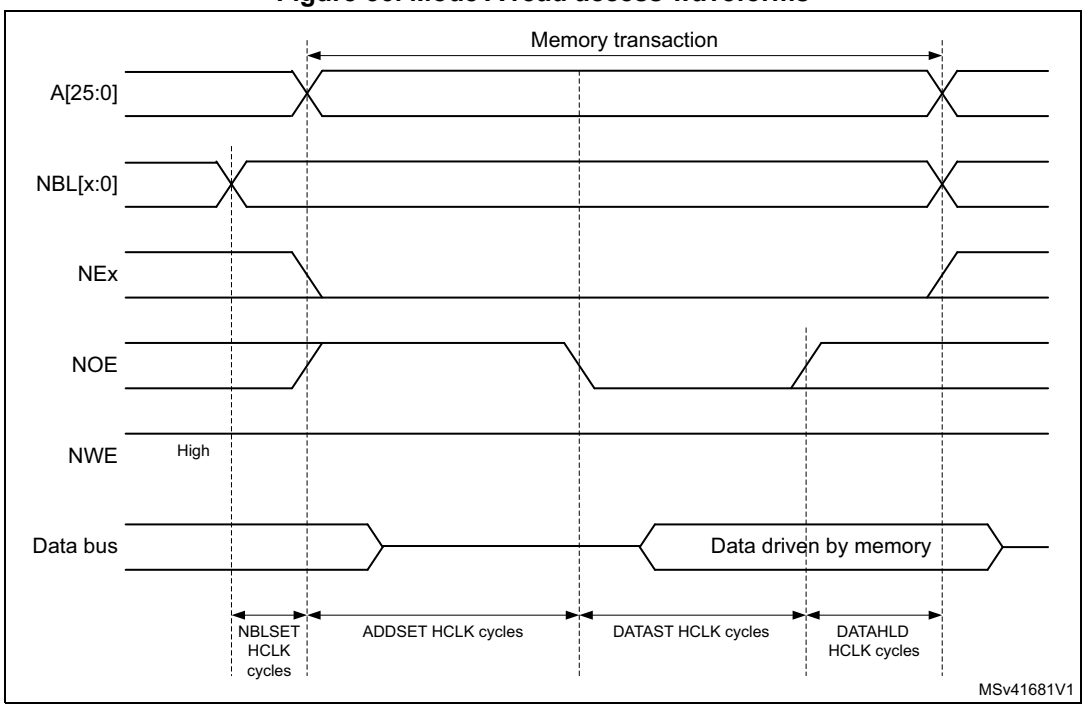
Bit number	Bit name	Value to set
3:2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 126. 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.

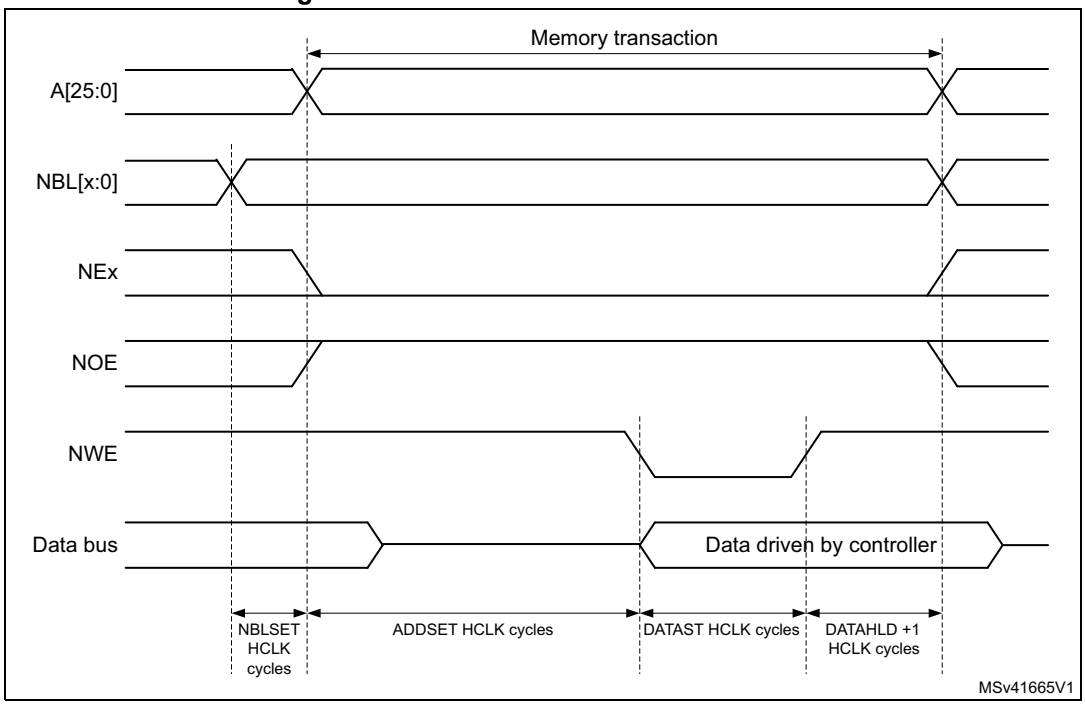
Mode A - SRAM/FRAM/PSRAM (CRAM) OE toggling

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 127. 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 127. 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 128. 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 129. 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

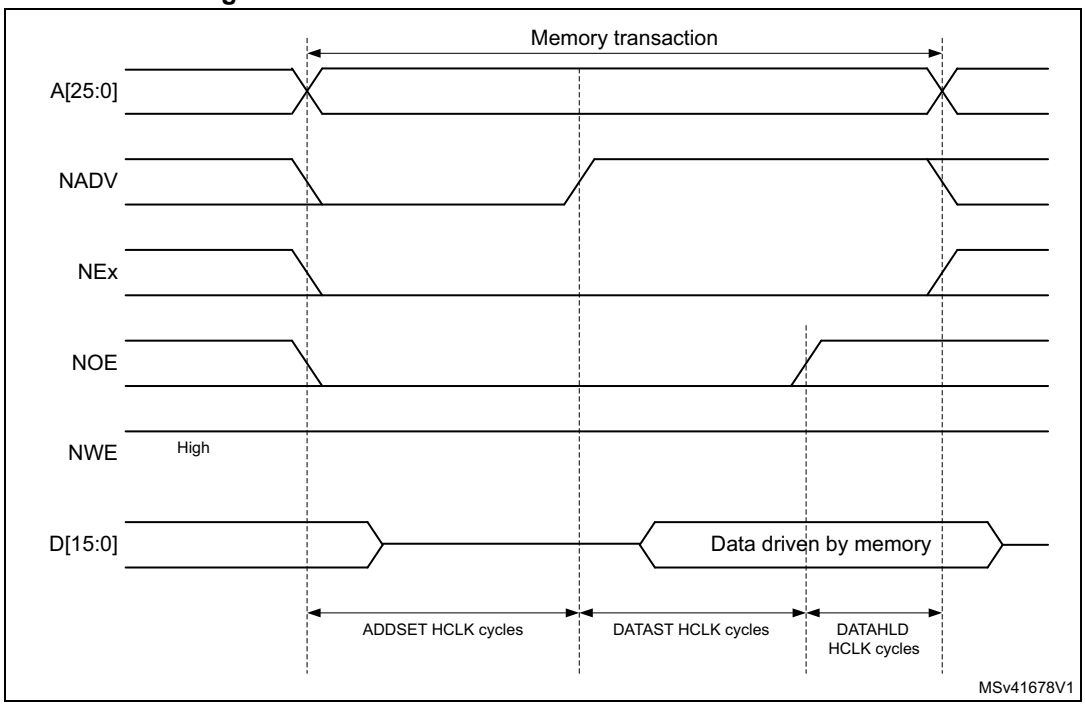


Figure 59. Mode 2 write access waveforms

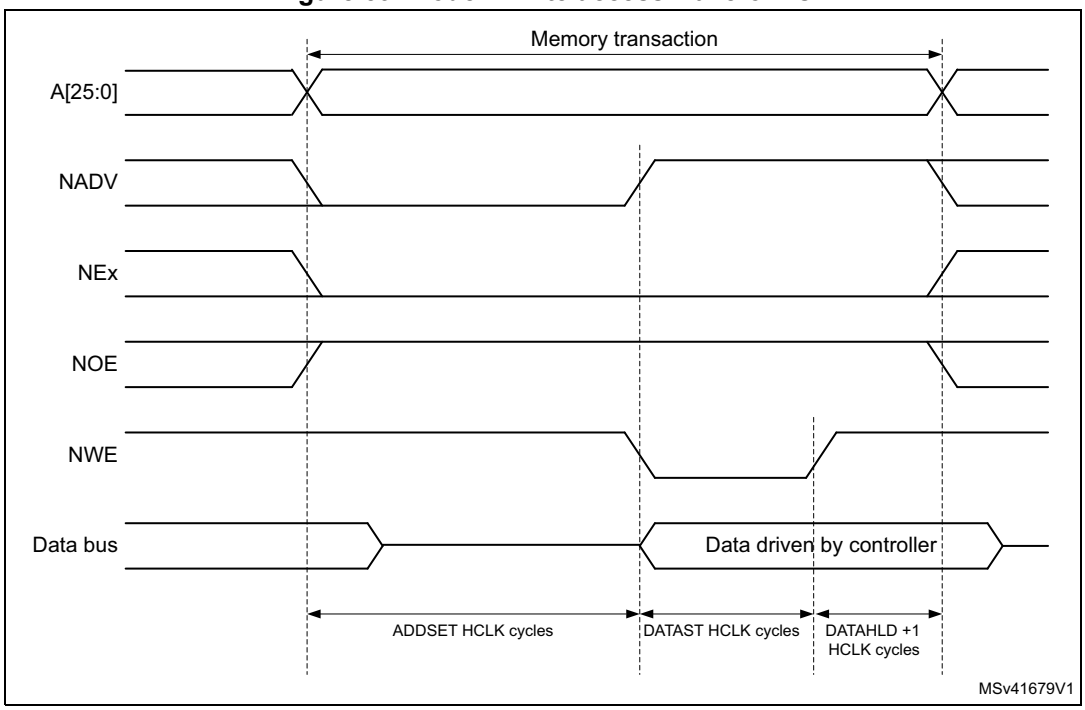
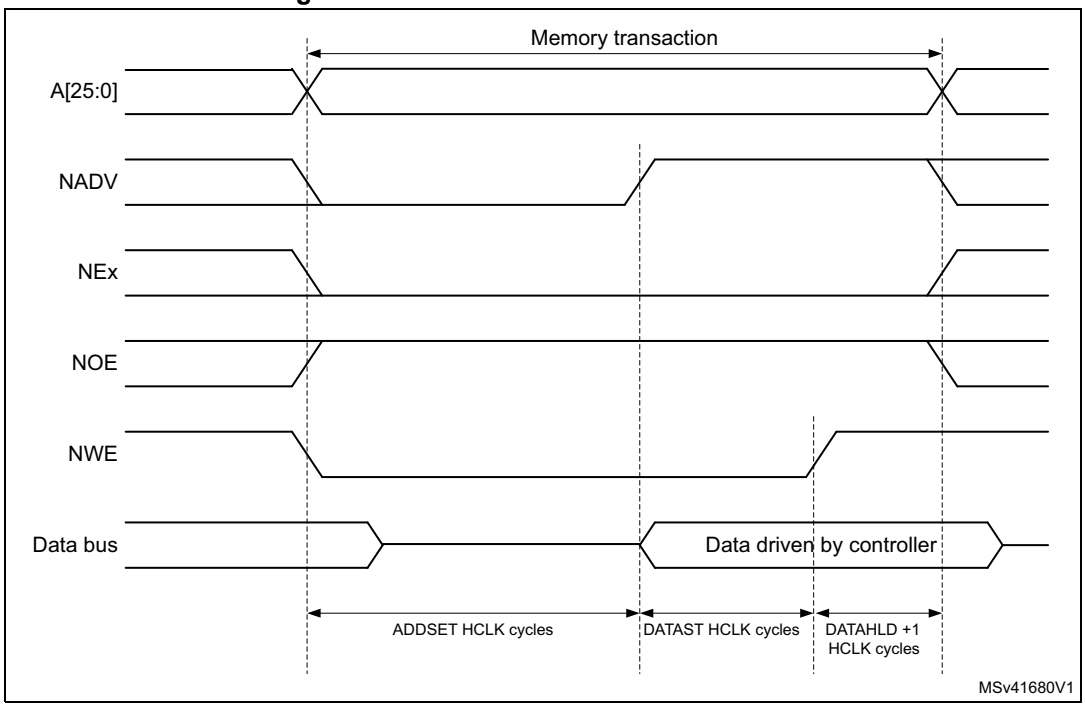


Figure 60. Mode B write access waveforms



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 130. 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 130. 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 131. 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 132. 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

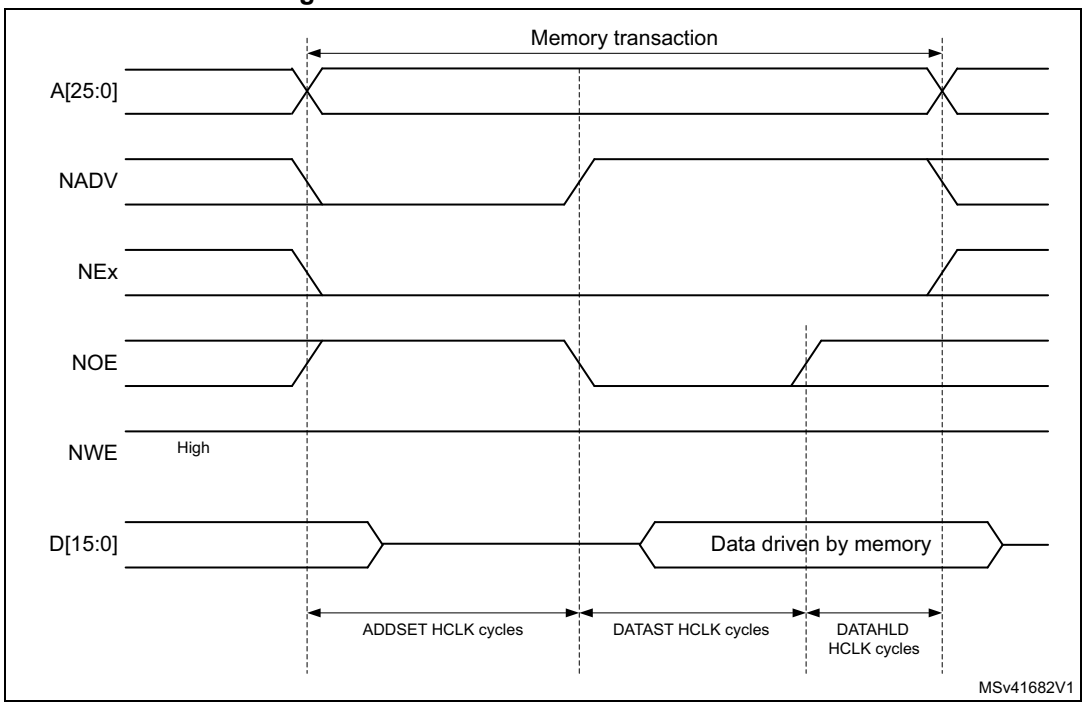
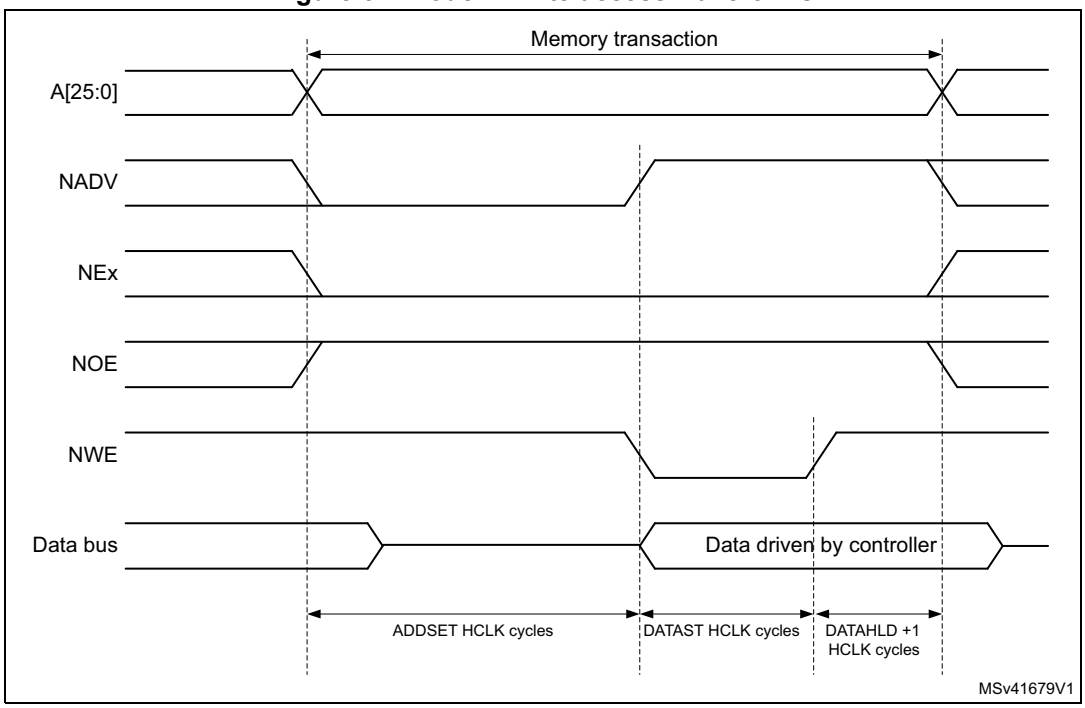


Figure 62. Mode C write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 133. 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 134. 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 135. 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

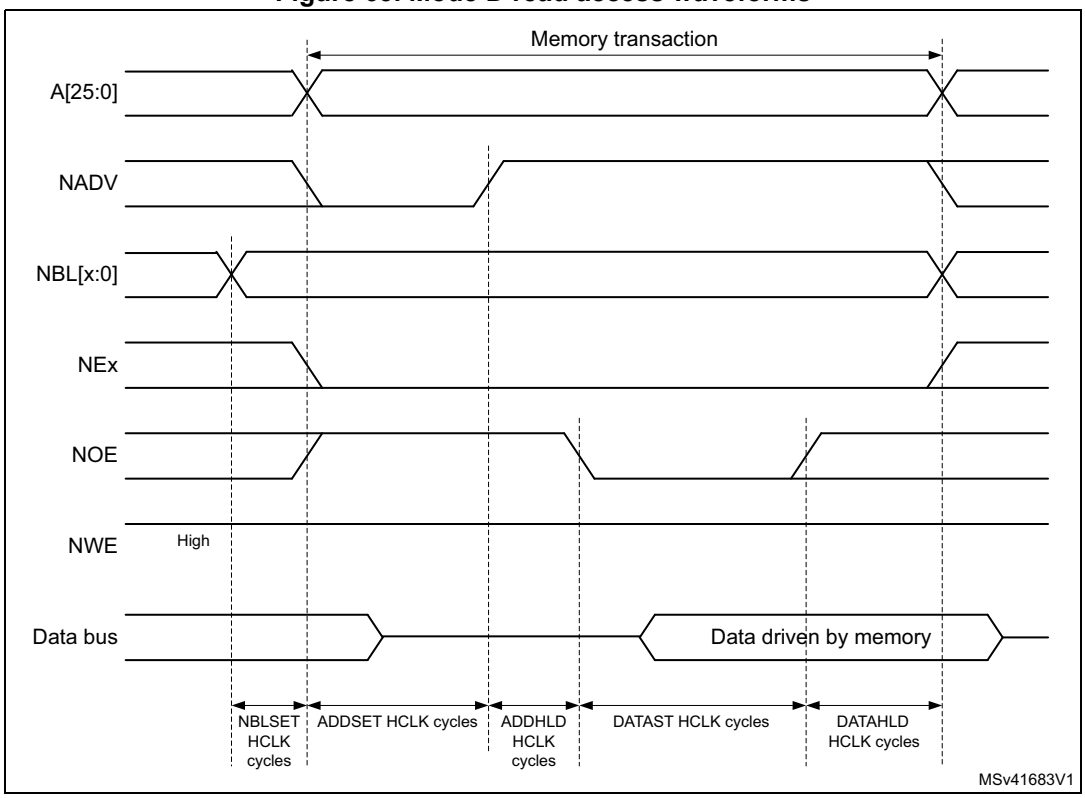
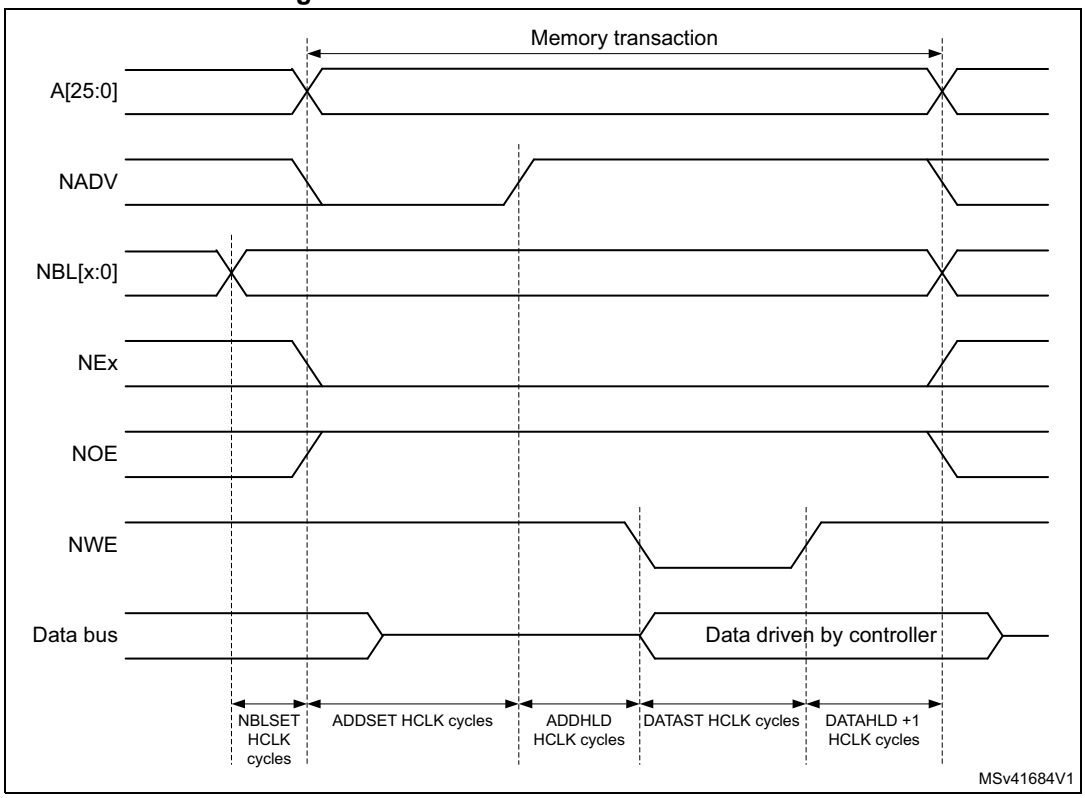


Figure 64. Mode D write access waveforms



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 136. 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 136. 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 137. 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 138. 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

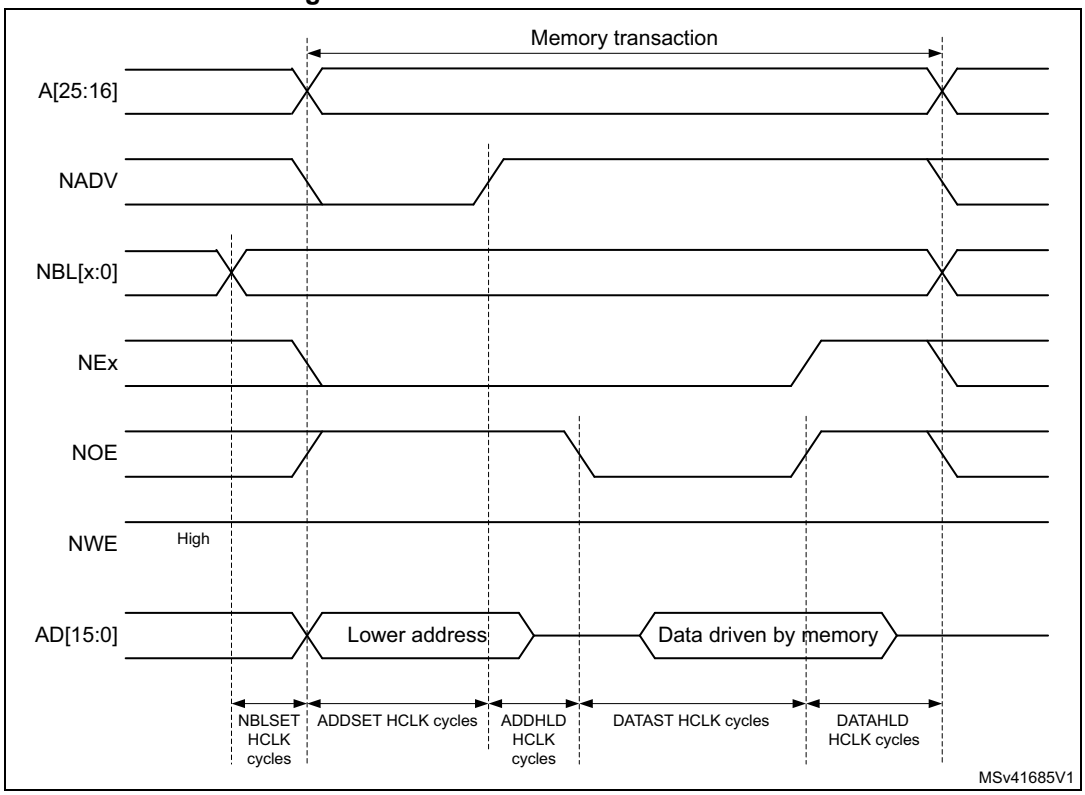
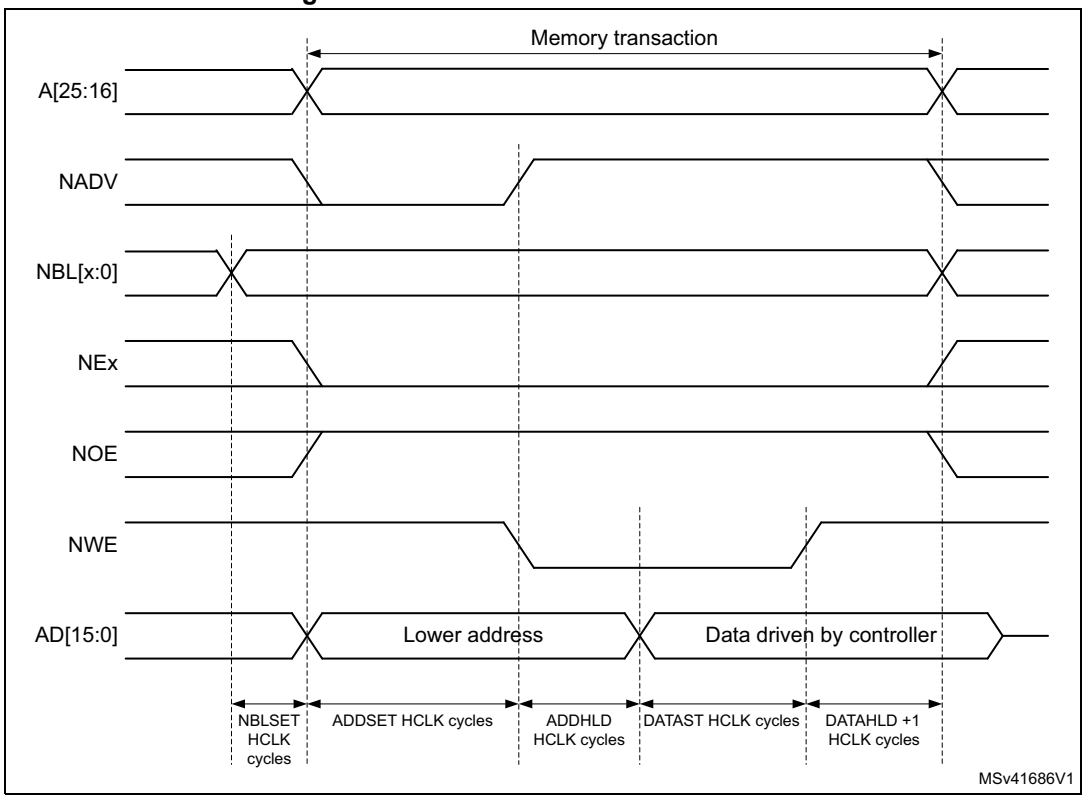


Figure 66. Muxed write access waveforms



The difference with mode D is the drive of the lower address byte(s) on the data bus.

Table 139. 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 139. 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 140. 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:

$$DATAST \geq (4 \times HCLK) + max\_wait\_assertion\_time$$

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   if

$$max\_wait\_assertion\_time > address\_phase + hold\_phase$$

   then:

$$DATAST \geq (4 \times HCLK) + (max\_wait\_assertion\_time - address\_phase - hold\_phase)$$

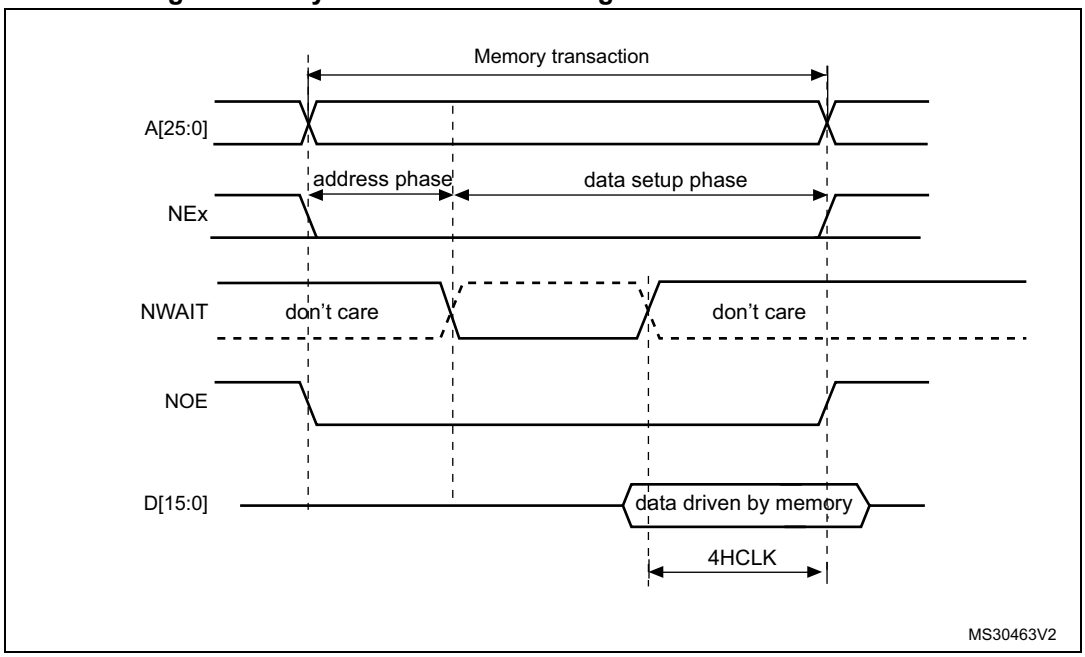
   otherwise

$$DATAST \geq 4 \times HCLK$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

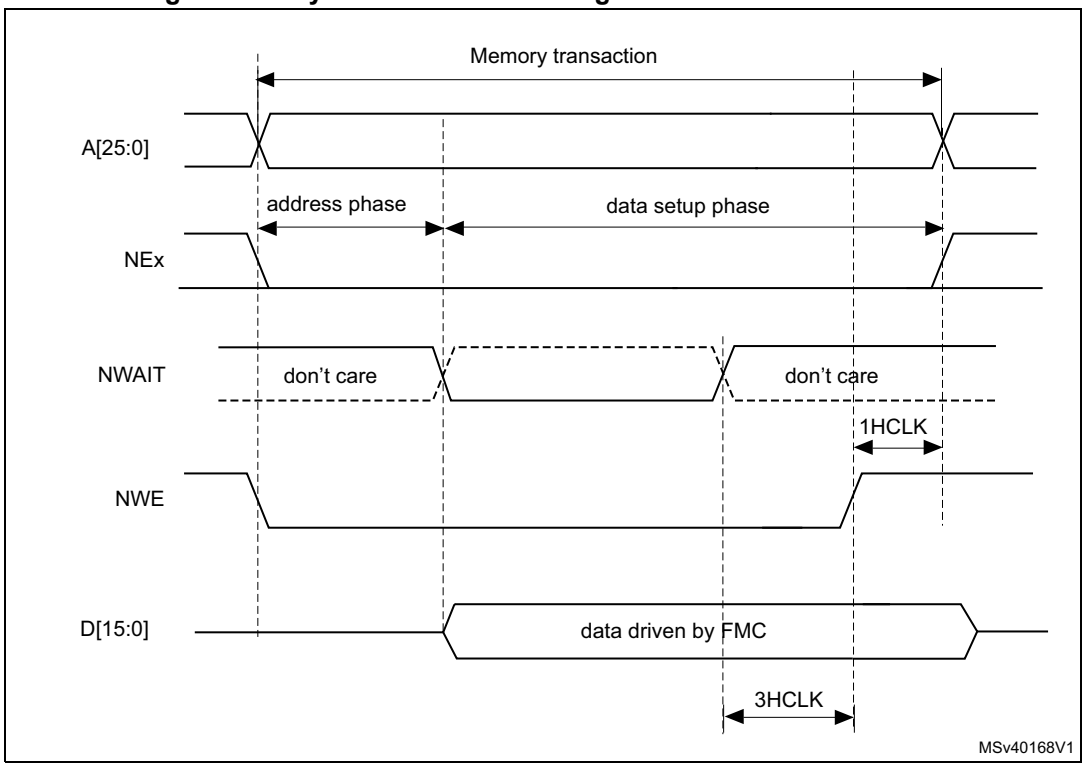
Figure 67 and *Figure 68* show the number of HCLK clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

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

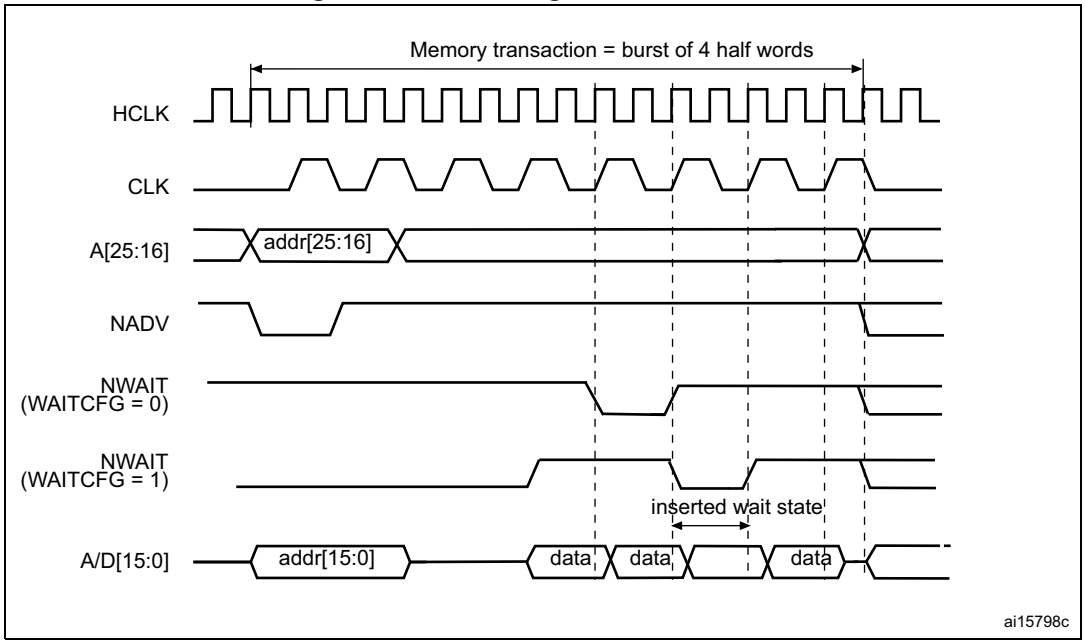
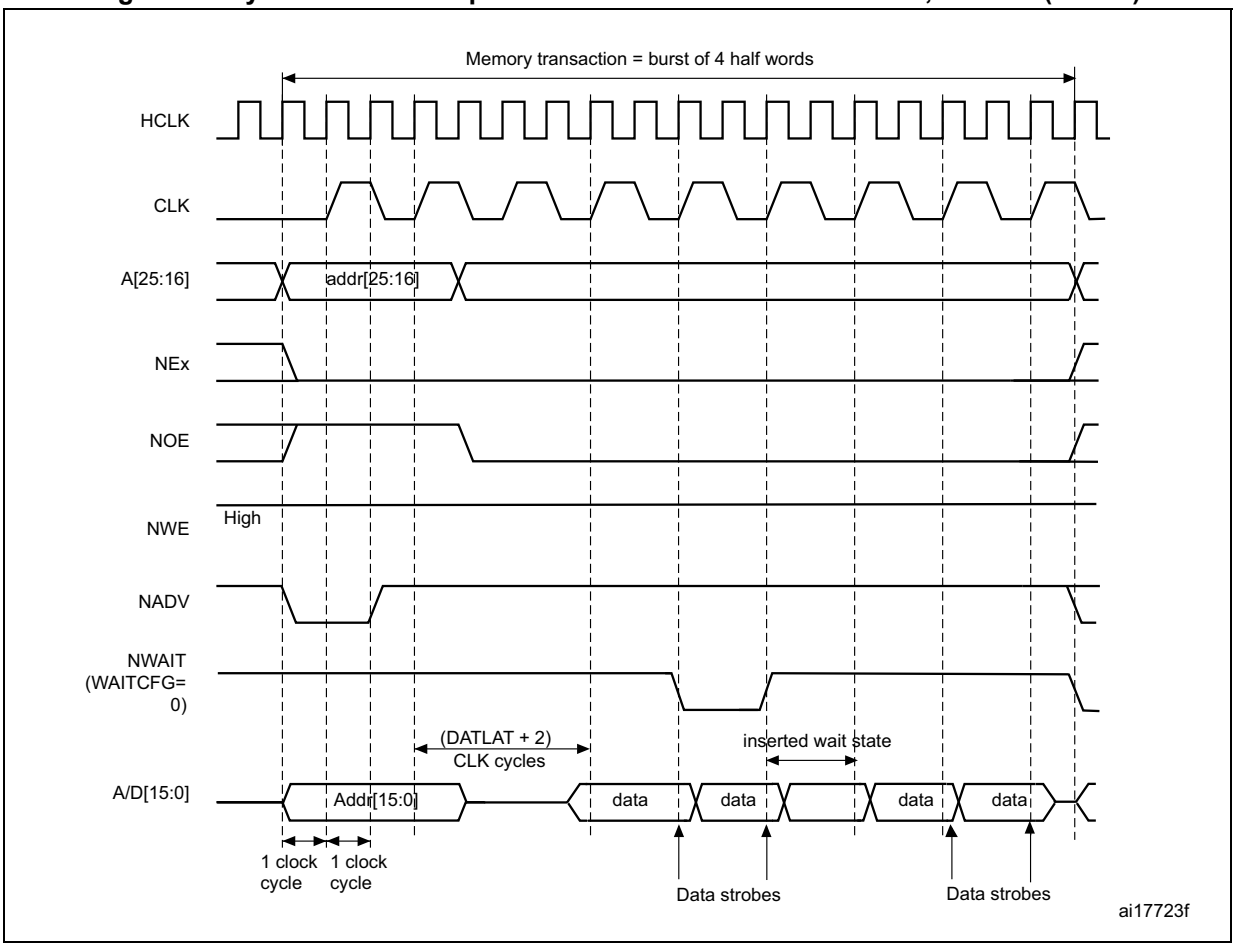


Figure 70. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)



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 141. 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

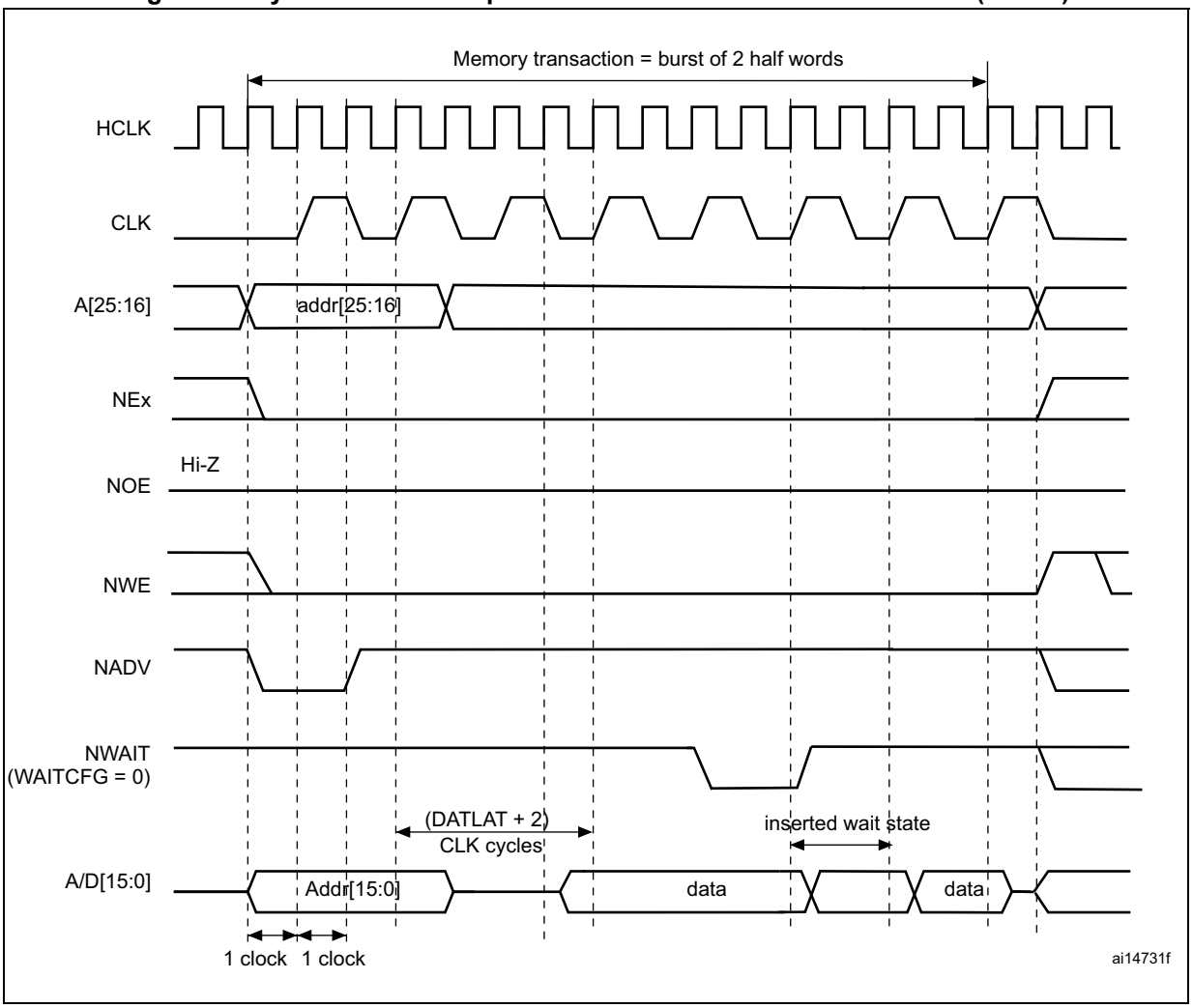
Table 141. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

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 142. 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 143. 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 143. 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 144. 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 145*) and ECC configuration.

Table 145. 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 145. 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 146. 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 147. 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 148 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND Flash controller are shown in gray.

Table 148. 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

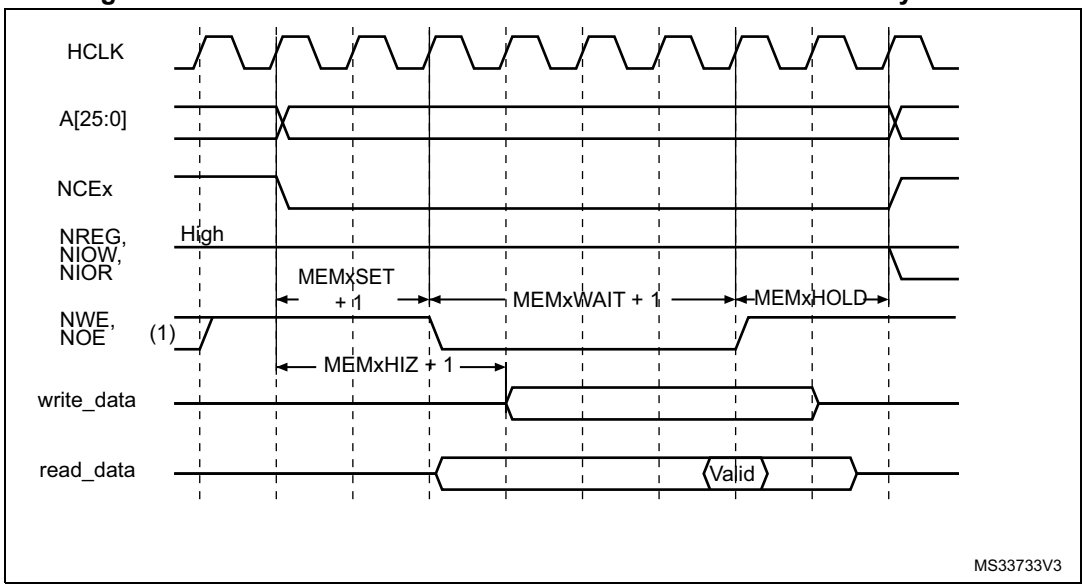
19.7.3 Timing diagrams for NAND Flash memory

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

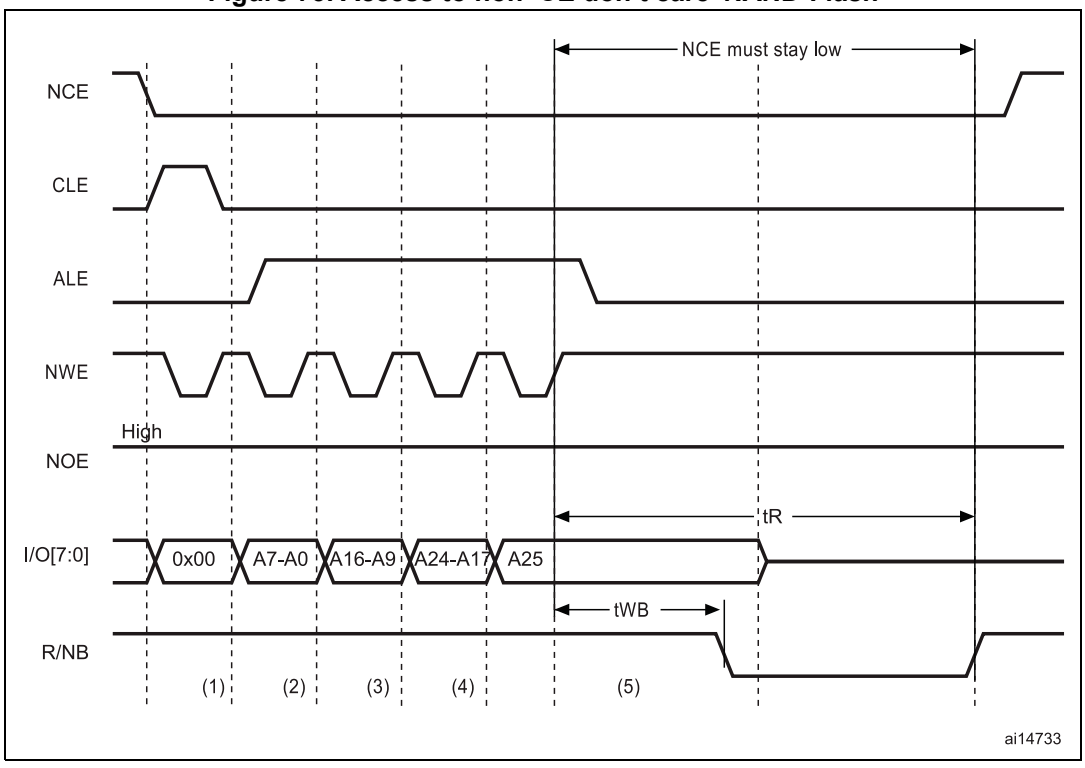
to implement the prewait functionality needed by some NAND Flash memories (see details in *Section 19.7.5: NAND Flash prewait functionality*).

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 149* describes the contents of these bitfields.

Table 149. 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 150. 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.	NBL SET [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.	NBL SET [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.	NBL SET [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.	NBL SET [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 150. 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	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 2.3 on page 87* for the register boundary addresses.

20 Octo-SPI interface (OCTOSPI)

20.1 Introduction

The OCTOSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR Flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- Indirect mode: all the operations are performed using the OCTOSPI registers to preset commands, addresses, data and transfer parameters.
- Automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting.
- Memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The OCTOSPI supports the following protocols with associated frame formats:

- the Regular-command frame format with the command, address, alternate byte, dummy cycles and data phase
- the HyperBus™ frame format

20.2 OCTOSPI main features

- Functional modes: Indirect, Automatic status-polling, and Memory-mapped
- Read and write support in Memory-mapped mode
- External (P)SRAM memory support
- Support for single, dual, quad and octal communication
- Dual-quad configuration, where eight bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- Asynchronous bus clock versus kernel clock support
- 8-, 16-, and 32-bit data accesses allowed
- DMA channel for Indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- AHB interface with transaction acceptance limited to one: the interface accepts the next transfer on AHB bus only once the previous is completed on memory side.

20.3 OCTOSPI implementation

Table 151. OCTOSPI implementation

OCTOSPI feature	OCTOSPI1
HyperBus standard compliant	X
Xcella standard compliant	X
XSPI (JEDEC251ES) standard compliant	X
AMBA® AHB compliant data interface	X
Asynchronous AHB clock versus kernel clock	X
Functional modes: Indirect, Automatic status-polling, and Memory-mapped	X
Read and write support in Memory-mapped mode	X
Dual-quad configuration	X
SDR (single-data rate) and DTR (double-transfer rate)	X
Data strobe (DS,DQS)	X
Fully programmable opcode	X
Fully programmable frame format	X
Integrated FIFO for reception and transmission	X
8-, 16-, and 32-bit data accesses	X
Interrupt on FIFO threshold, timeout, operation complete, and access error	X
Extended CSHT timeout	-
Memory-mapped write	X
Refresh counter	X

20.4 OCTOSPI functional description

20.4.1 OCTOSPI block diagram

Figure 74. OCTOSPI block diagram in octal configuration

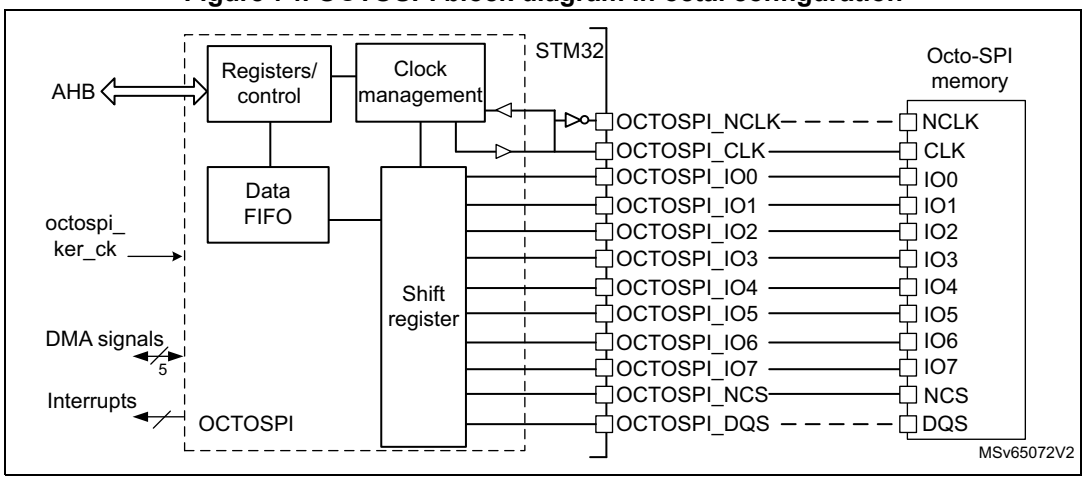


Figure 75. OCTOSPI block diagram in quad configuration

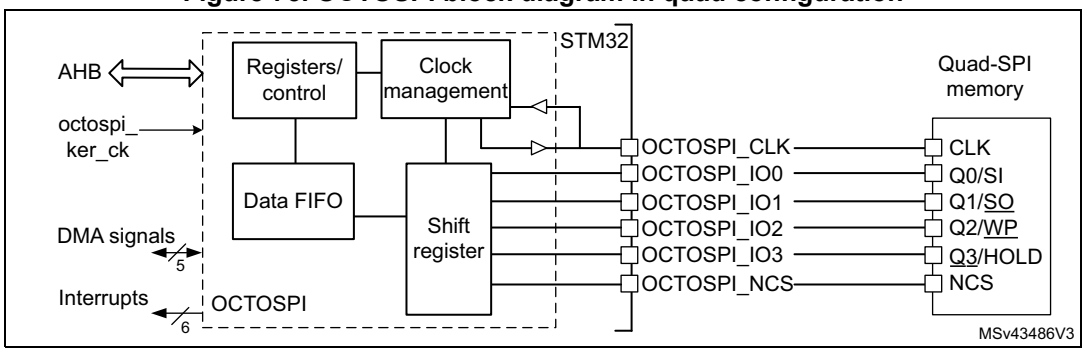
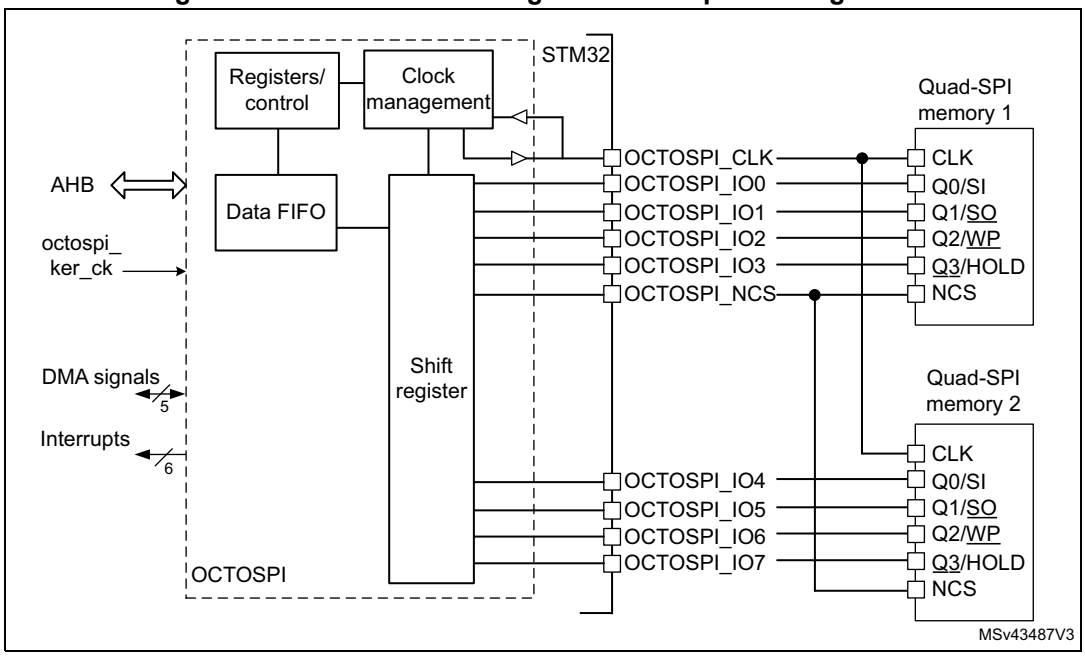


Figure 76. OCTOSPI block diagram in dual-quad configuration



20.4.2 OCTOSPI interface to memory modes

The OCTOSPI supports the following protocols:

- Regular-command protocol
- HyperBus protocol

The OCTOSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select
- CLK: communication clock
- NCLK: inverted clock used only in the 1.8 V HyperBus protocol
- DQS: data strobe used only in Regular-command protocol as input only
- IO[3:0]: data bus LSB
- IO[7:4]: data bus MSB used in dual-quad and octal configurations

20.4.3 OCTOSPI Regular-command protocol

When in Regular-command protocol, the OCTOSPI communicates with the external device using commands. Each command can include the following phases:

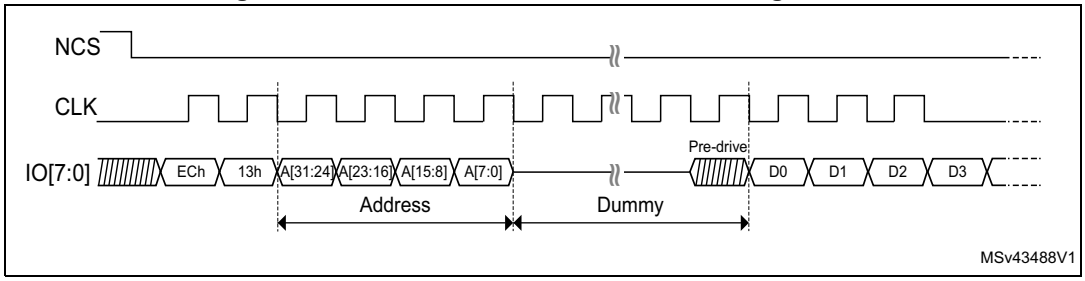
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped, but at least one of the instruction, address, alternate byte, or data phases must be present.

The NCS falls before the start of each command and rises again after each command finishes.

In Memory-mapped mode, both read and write operations are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

Figure 77. SDR read command in octal configuration



The specific Regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured in ISIZE[1:0] of OCTOSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of OCTOSPI_IR.

The instruction phase can optionally send:

- 1 bit at a time from the IO0/SO signal (Single-SPI mode)
- 2 bits at a time (over IO0/IO1 in Dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in Quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in Octal-SPI mode).

This can be configured using IMODE[2:0] of OCTOSPI_CCR.

The instruction can be sent in DTR (double-transfer rate) mode on each rising and falling edge of the clock, by setting IDTR in OCTOSPI_CCR.

When IMODE[2:0] = 000 in OCTOSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

In Memory-mapped mode, the instruction used for the write operation is specified in OCTOSPI_WIR and the instruction format is specified in OCTOSPI_WCCR. The instruction used for the read operation and the instruction format are specified in OCTOSPI_IR and OCTOSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured in ADSIZE[1:0] of OCTOSPI_CCR.

In Indirect and Automatic status-polling modes, the address bytes to be sent are specified in ADDRESS[31:0] of OCTOSPI_AR. In Memory-mapped mode, the address is given directly via the AHB (from any master in the system).

The address phase can send:

- 1 bit at a time (over SOI in Single-SPI mode)
- 2 bits at a time (over IO0/IO1 in Dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in Quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in Octal-SPI mode)

This can be configured using ADMODE[2:0] of OCTOSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR of OCTOSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In Memory-mapped mode, the address format for the write operation is specified in OCTOSPI_WCCR. The address format for the read operation is specified in OCTOSPI_CCR.

Alternate-bytes phase

In the alternate-bytes phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured in ABSIZE[1:0] of OCTOSPI_CCR. The bytes to be sent are specified in OCTOSPI_ABR.

The alternate-byte phase can send:

- 1 bit at a time (over SOI in Single-SPI mode)
- 2 bits at a time (over IO0/IO1 in Dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in Quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in Octal-SPI mode)

This can be configured using ABMODE[2:0] of OCTOSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR of OCTOSPI_CCR.

When ABMODE[2:0] = 000, 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 the Dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the Quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in OCTOSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 4:3 of ALTERNATE[31:0] while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In Memory-mapped mode, the alternate bytes used for the write operation are specified in OCTOSPI_WABR and the alternate byte format is specified in OCTOSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in OCTOSPI_ABR and OCTOSPI_CCR.

Dummy-cycle phase

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when the higher clock frequencies are used. The number of cycles given during this phase is specified in DCYC[4:0] of OCTOSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the Dual-SPI, the Quad-SPI or the Octal-SPI mode, to receive data from the external device.

In Memory-mapped mode, the dummy cycles for the write operations are specified in OCTOSPI_WTCR. The dummy cycles for the read operation are specified in OCTOSPI_TCR.

Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In Indirect mode, the number of bytes to be sent/received is specified in OCTOSPI_DLR. In this mode, the data to be sent to the external device must be written to OCTOSPI_DR, while in Indirect-read mode the data received from the external device is obtained by reading OCTOSPI_DR.

In Automatic status-polling mode, the number of bytes to be received is specified in OCTOSPI_DLR and the data received from the external device can be obtained by reading OCTOSPI_DR.

In Memory-mapped mode, the data read or written, is sent or received directly over the AHB to the Cortex core 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)
- 4 bits at a time (over IO0 to IO3 in Quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in Octal-SPI mode)

This can be configured using DMODE[2:0] of OCTOSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR of OCTOSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in Indirect-write mode.

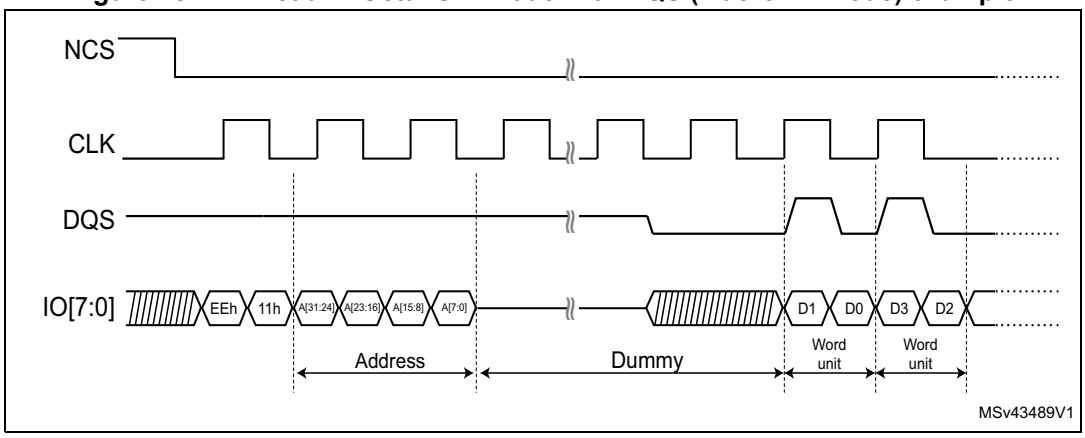
In Memory-mapped mode, the data format for the write operation is specified in OCTOSPI_WCCR. The data format for the read operation is specified in OCTOSPI_CCR.

DQS usage

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE of OCTOSPI_CCR.

Figure 78. DTR read in Octal-SPI mode with DQS (Macronix mode) example



20.4.4 OCTOSPI Regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (whose I/Os are shared with IO0). The data received from the external device arrives via SI (whose I/Os are shared with IO1).

The different phases can each be configured separately to use this Single-bit mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in Single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0 (to deactivate the “write protect” function).
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

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 to 010 the IMODE,ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in Dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the 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.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 010, 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 the Quad-SPI mode by setting to 011 the IMODE,ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in Quad-SPI mode:

- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 011, IO0 to IO3 are all high-impedance.

IO2 and IO3 are used only in Quad-SPI mode. If none of the phases are configured to use the Quad-SPI mode, then the pins corresponding to IO2 and IO3 can be used for other functions even while the OCTOSPI is active.

Octal-SPI mode

In regular Octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the Octal-SPI mode by setting to 100 the IMODE,ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase that is configured in Octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE[2:0] = 100, IO[0:7] are all high-impedance.

IO[4:7] are used only in Octal-SPI mode. If none of the phases are configured to use Octal-SPI mode, then the pins corresponding to IO[4:7] can be used for other functions even while the OCTOSPI is active.

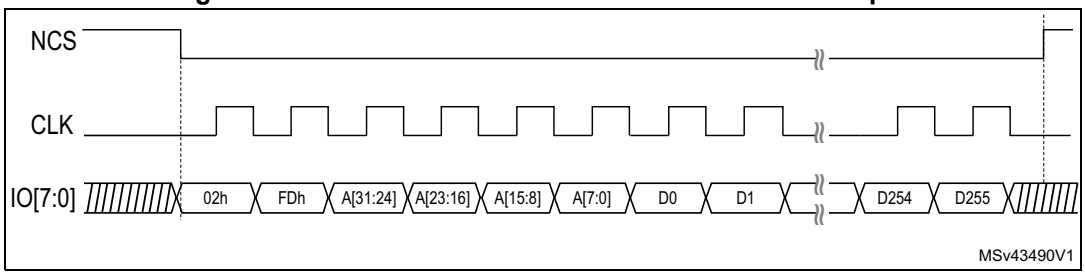
Single-data rate (SDR) mode

By default, all the phases operate in Single-data rate (SDR) mode.

In SDR mode, when the OCTOSPI drives the IO0/SO, IO1 to IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the OCTOSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in OCTOSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 79. SDR write command in Octo-SPI mode example.



Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in OCTOSPI_CCR.

In Memory-mapped mode, the DTR mode for each phase of the write operations is specified in OCTOSPI_WCCR. The DTR mode for each phase of the read operations is specified in OCTOSPI_CCR.

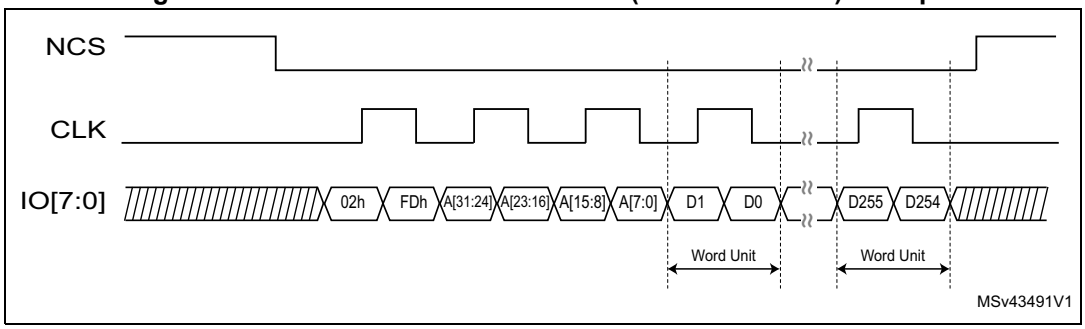
In DTR mode, when the OCTOSPI drives the IO0/SO and IO1to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the OCTOSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in OCTOSPI_CCR, the software must clear SSHIFT in OCTOSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set DHQC of OCTOSPI_TCR, to shift the outputs by a quarter of cycle and avoid to hold issues on the memory side.

Note: DHQC must not be set when the prescaler value is 0, as this action leads to unpredictable behavior.

Figure 80. DTR write in Octal-SPI mode (Macronix mode) example



Dual-quad configuration

When DMM = 1 in OCTOSPI_CR, the OCTOSPI is in dual-memory configuration: if DMODE = 100,two external Quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the Single-SPI, Dual-SPI, and Quad-SPI modes, as well as with either the SDR or DTR mode.

The device size, as specified in DEVSIZE[4:0] of OCTOSPI_DCR1, must reflect the total external device capacity, that is the double of the size of one individual component.

If address X is even, then the byte that the OCTOSPI gives for address X is the byte at the address X/2 of device A, and the byte that the OCTOSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in Regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the OCTOSPI must be configured with a data length of 2 bytes (16 bits), and the OCTOSPI receives one byte from each device.

If each device gives a status of 16 bits, then the OCTOSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The forth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in OCTOSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

20.4.5 HyperBus protocol

The OCTOSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins depending on the operating voltage:

- IO[7:0] as bidirectional data bus
- RWDS for read and write data strobe and latency insertion (mapped on DQS pin)
- NCS
- CLK
- NCLK for 1.8 V operations

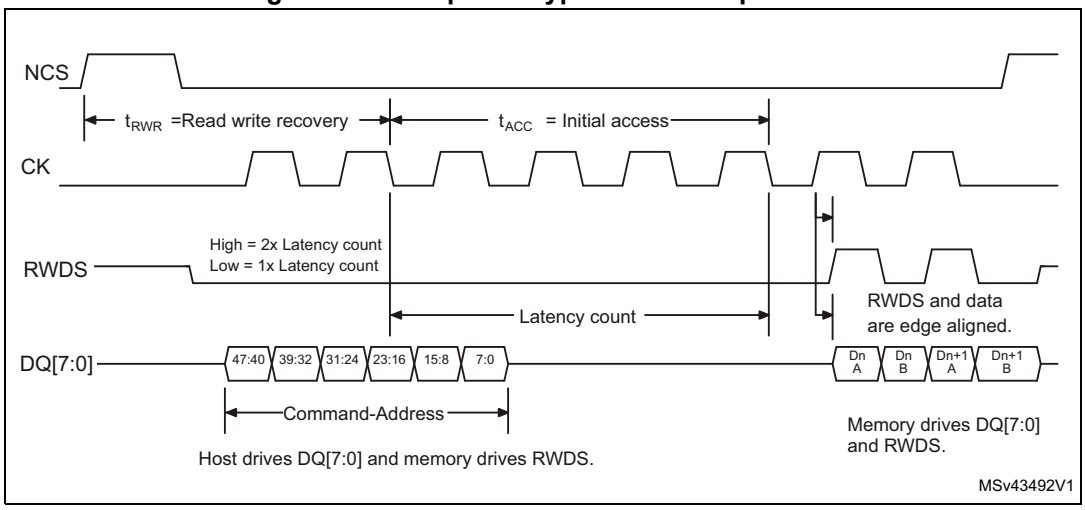
The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- Command/address phase
- Data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

Figure 81. Example of HyperBus read operation



The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

Command/address phase

During this initial phase, the OCTOSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

Table 152. Command/address phase description

CA bit	Bit name	Description
47	R/W#	Identifies the transaction as a read or a write.
46	Address space	Indicates if the transaction accesses the memory or the register space.
45	Burst type	Indicates if the burst is linear or wrapped.
44-16	Row and upper column address	Selects the row and the upper column addresses.
15-3	Reserved	-
2-0	Lower column address	Selects the starting 16-bit word within the half page.

The address space is configured through the memory type MTYP[2:0] of OCTOSPI_DCR1.

The total size of the device is configured in DEVSIZE[4:0] of OCTOSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

Read/write operation with initial latency

The HyperBus read and write operations need to respect two timings:

- t_{RWR}: minimal read/write recovery time for the device (defined in TRWR[7:0] of OCTOSPI_HLCR)
- t_{ACC}: access time for the device (defined in TAC[7:0] of OCTOSPI_HLCR)

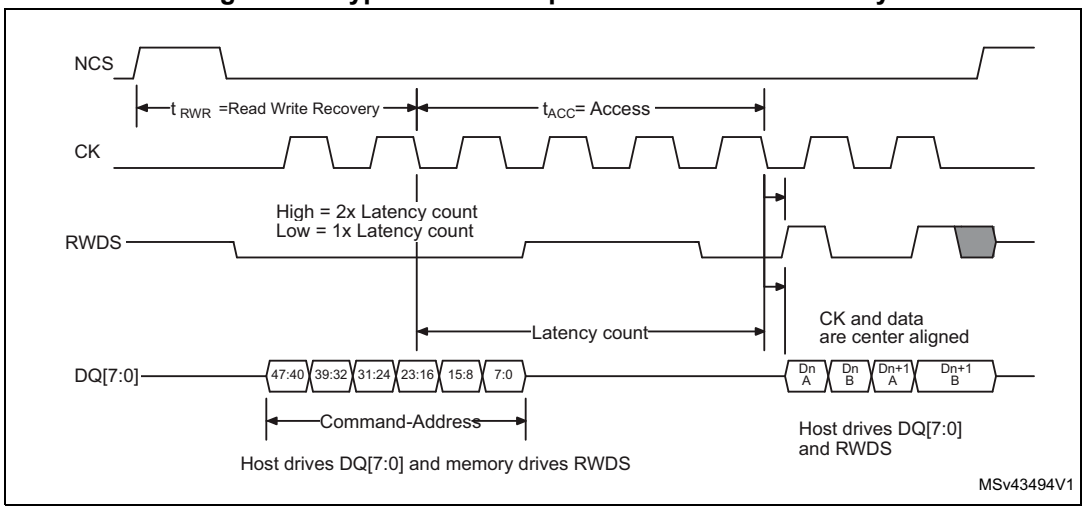
During the read operation, the RWDS is used by the device, in two ways (see *Figure 81*):

- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

During the write operation the RWDS is used:

- by the device, during the command/address phase, to request an additional latency.
- by the OCTOSPI, during the data phase, for write data masking.

Figure 82. HyperBus write operation with initial latency



Read/write operation with additional latency

If the device needs an additional latency (during refresh period of a SDRAM for example), RWDS must be tied to one during one of the RWDS signals, during the command/address phase.

An additional t_{ACC} duration is added by the OCTOSPI to meet the device request.

Figure 83. HyperBus read operation with additional latency

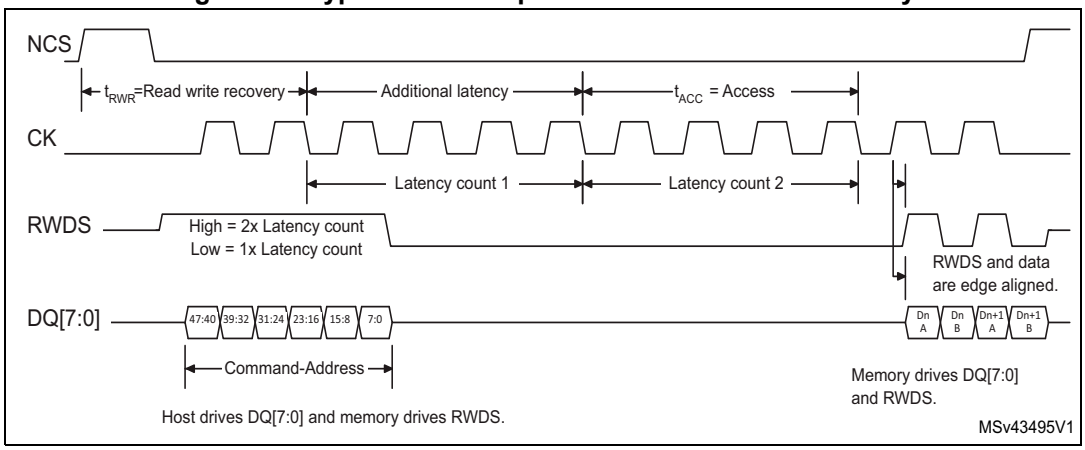
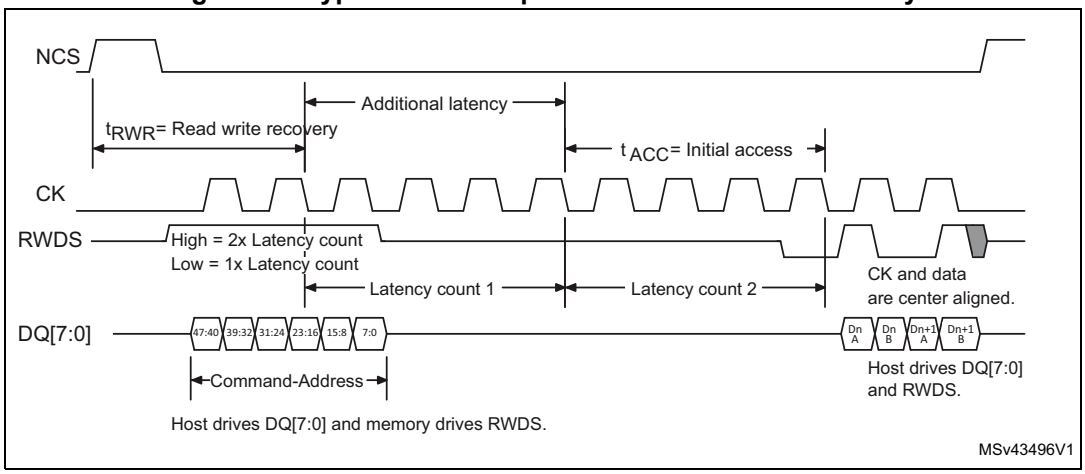


Figure 84. HyperBus write operation with additional latency



Fixed-latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

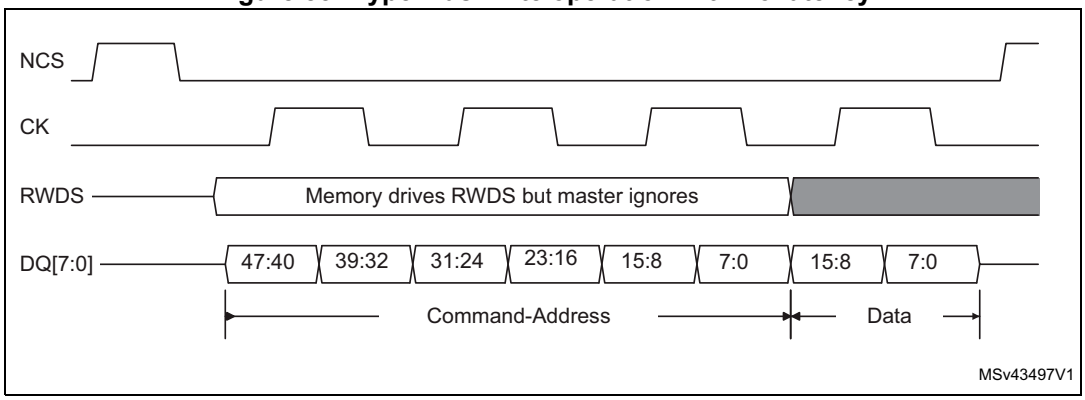
The latency can be forced to 2 x t_{ACC} by setting LM of OCTOSPI_HLCR.

In this OCTOSPI latency mode, the state of the RWDS signal is not taken into account by the OCTOSPI and an additional latency is always added, leading to a fixed 2 x t_{ACC} latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in OCTOSPI_HLCR.

Figure 85. HyperBus write operation with no latency

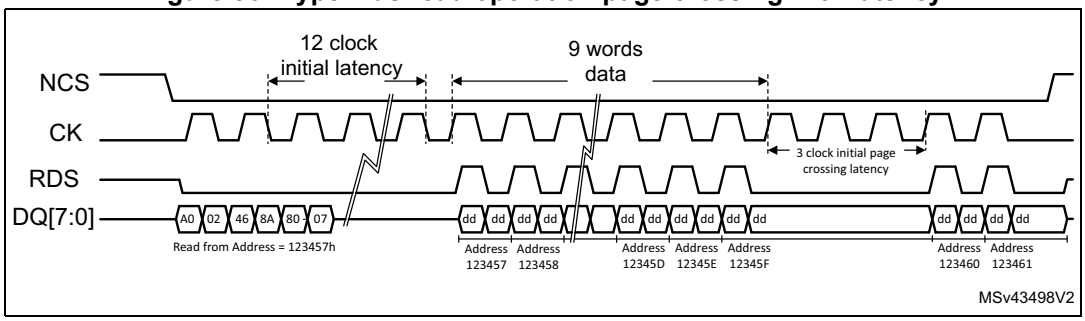


Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the t_{ACC} time.

Figure 86. HyperBus read operation page crossing with latency



20.4.6 Specific features

The OCTOSPI supports some specific features, such as:

- Wrap support
- NCS boundary and refresh

Wrap support

The OCTOSPI supports an hybrid wrap as defined by the HyperBus protocol. An hybrid wrap is also supported in the Regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in OCTOSPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 4 bytes. Wrapped reads are supported for both HyperBus and Regular-command protocols. To enable wrapped-read accesses, the dedicated registers OCTOSPI_WPxxx must be programmed according to the wrapped-read access characteristics. The dedicated OCTOSPI_WPxxx registers apply for both HyperBus and Regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

NCS boundary and refresh

Two processes can be activated to regulate the OCTOSPI transactions:

- NCS boundary
- Refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with, is configured in CSBOUND[4:0] of OCTOSPI_DCR3 and it is equal to $2^{CSBOUND}$.

As an example, if CSBOUND(4:0] = 0x4, the boundary is set to 2^4 = 16 bytes. As a consequence, the NCS is released each time that the LSB address is equal to 0xF and each time that a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled and a minimum value of 3 is recommended.

The NCS boundary feature cannot be used for Flash memory devices in write mode since a command is necessary to program another page of the Flash memory.

The refresh feature limits the duration of the transactions to the value programmed in REFRESH[31:0] of OCTOSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

Re-starting after an interrupted transfer

When a read or write operation is interrupted by a timeout or communication regulation feature, the Octo-SPI interface, as soon as possible after getting back the port ownership, re-issues the initial command sequence together with the address following the last address actually accessed before interruption. The transfer initially set goes on and ends seamlessly.

20.4.7 OCTOSPI operating modes introduction

The OCTOSPI has the following operating modes regardless of the low-level protocol used (either Regular-command or HyperBus):

- Indirect mode (read or write)
- Automatic status-polling mode
- Memory-mapped mode

20.4.8 OCTOSPI Indirect mode

In Indirect mode, the commands are started by writing to the OCTOSPI registers and the data is transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 0 in OCTOSPI_CR, the OCTOSPI is in Indirect-write mode: bytes are sent to the external device during the data phase. Data is provided by writing to OCTOSPI_DR.

When FMODE[1:0] = 01, the OCTOSPI is in Indirect-read mode: bytes are received from the external device during the data phase. Data is recovered by reading OCTOSPI_DR.

In Indirect mode, when the OCTOSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The OCTOSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access

The number of bytes to be read/written is specified in OCTOSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the OCTOSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in OCTOSPI_CCR.
- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the OCTOSPI 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 bit is set in OCTOSPI_SR and an interrupt is generated if TCIE = 1 in OCTOSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in OCTOSPI_DCR1.

Triggering the start of a transfer in Regular-command protocol

Depending on the OCTOSPI configuration, there are three different ways to trigger the start of a transfer in Indirect mode when using Regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in Indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data needs to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in OCTOSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data needs to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in OCTOSPI_AR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if data needs to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in OCTOSPI_DR

A write to OCTOSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in OCTOSPI_SR.

Triggering the start of a transfer in HyperBus protocol

Depending on the OCTOSPI configuration, there are different ways to trigger the start of a command in Indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication in Indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (OCTOSPI_AR) in Indirect-read mode (when FMODE = 01).
- when a write is performed to DATA[31:0] (OCTOSPI_DR) in Indirect-write mode (when FMODE = 00).
- when a write is performed to INSTRUCTION[31:0] (OCTOSPI_IR) for both Indirect read and write modes

Note: In case of HyperBus, a (dummy) write to OCTOSPI_IR is required to trigger the transfer, as for Regular-command protocol.

As soon as a transfer is started, the BUSY bit (OCTOSPI_SR[5]) is automatically set.

FIFO and data management

Data in Indirect mode passes through a 32-byte FIFO that is internal to the OCTOSPI. FLEVEL in OCTOSPI_SR indicates how many bytes are currently being held in the FIFO.

AHB burst transactions are supported. Data of the burst are successively written in OCTOSPI_DR and immediately transferred in the internal FIFO.

In Indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in OCTOSPI_DR. A word write adds 4 bytes to the FIFO, an half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to OCTOSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets 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 is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In Indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

20.4.9 OCTOSPI Automatic status-polling mode

In Automatic status-polling mode, the OCTOSPI periodically starts a command to read a defined number of status bytes (up to four). 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 access to the device begins in the same manner as in Indirect-read mode. BUSY in OCTOSPI_SR goes high at this point and stays high even between the periodic accesses.

The content of MASK[31:0] in OCTOSPI_PSMAR is used to mask the data from the external device in Automatic status-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] in OCTOSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in OCTOSPI_CR, the AND-match mode is activated: SMF is set in OCTOSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in OCTOSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in OCTOSPI_CR, the 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 until the OCTOSPI is disabled (EN = 0).

OCTOSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in OCTOSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In Automatic status-polling mode, variable latency is not supported. As a consequence, the memory must be configured in fixed latency.

20.4.10 OCTOSPI Memory-mapped mode

When configured in Memory-mapped mode, the external SPI device is seen as an internal memory.

Note: More than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256 Mbytes range, then an AHB error is given. The effect of this error depends on the AHB master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the OCTOSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations ends up quickly since their results were pre-fetched.

By default, the OCTOSPI never stops its prefetch operation, it either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time.

Since external devices tend to consume more when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in OCTOSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in OCTOSPI_LPTR, when x cycles have elapsed without an access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

20.4.11 OCTOSPI configuration introduction

The OCTOSPI configuration is done in three steps:

1. OCTOSPI system configuration
2. OCTOSPI device configuration
3. OCTOSPI mode configuration

20.4.12 OCTOSPI system configuration

The OCTOSPI is configured using OCTOSPI_CR. The user must program:

- Functional mode with FMODE[1:0]
- Automatic status-polling mode behavior if needed with PMM and APMS
- FIFO level with FTHRES
- DMA usage with DMAEN
- Timeout counter usage with TCEN
- Dual-quad configuration, if needed, with DMM (only for Quad-SPI mode)

In case of an interrupt usage, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in OCTOSPI_LPTR.

The DMA channel must not be enabled during the OCTOSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and OCTOSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

20.4.13 OCTOSPI device configuration

The parameters related to the external device targeted are configured through OCTOSPI_DCR1 and OCTOSPI_DCR2.The user must program:

- Device size with DEVSIZE[4:0]
- Chip-select minimum high time with CSHT[2:0]
- Clock mode with FRCK and CKMODE
- Device frequency with PRESCALER[7:0]

MTYP[2:0] defines the memory type to be used for 8-line modes:

- Micron mode with D0/D1 ordering in 8-data-bit mode (DMODE[2:0] = 100)
- Macronix mode with D1/D0 ordering in 8-data-bit mode (DMODE[2:0] = 100)
- HyperBus memory mode: the protocol follows the HyperBus specification, and an 8-data-bit DTR mode must be selected.
- HyperBus register mode, addressing register space: the memory-mapped accesses in this mode must be non-cacheable, or the Indirect read/write modes must be used.

DEVSIZE[4:0] defines the size of external memory using the following formula:

$$\text{Number of bytes in the device} = 2^{[\text{DEVSIZE}+1]}$$

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in Indirect mode, but the addressable space in Memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] indicates the total capacity of the two devices together.

When the OCTOSPI 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 external device requires more time between commands, the chip-select high time CSHT[2:0] can be used to specify the minimum number of CLK cycles for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (t_{ACC} and t_{RWR}) and the Latency mode must be configured in OCTOSPI_HLCR.

20.4.14 OCTOSPI Regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the Indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the Indirect-read mode is selected and data can be read from the external device.

When the OCTOSPI 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 OCTOSPI_DLR.
2. Specify the frame timing in OCTOSPI_TCR.
3. Specify the frame format in OCTOSPI_CCR.
4. Specify the instruction in OCTOSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
6. Specify the targeted address in OCTOSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

If neither the address register (OCTOSPI_AR) nor the data register (OCTOSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in Indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

Automatic status-polling mode configuration

The Automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and the data is retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in OCTOSPI_DLR, it is ignored and only 4 bytes are read.The periodicity is specified in OCTOSPI_PIR.

Once the status data has been retrieved, the following can be processed:

- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in OCTOSPI_PSMKR, and can be ORed or ANDed with the value stored in OCTOSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled; The OCTOSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in OCTOSPI_DR.

When the OCTOSPI is used in Automatic status-polling mode, the frames are constructed in the following way:

1. Specify the input mask in OCTOSPI_PSMKR.
2. Specify the comparison value in OCTOSPI_PSMAR.
3. Specify the read period in OCTOSPI_PIR.
4. Specify a number of data bytes to read in OCTOSPI_DLR.
5. Specify the frame timing in OCTOSPI_TCR.
6. Specify the frame format in OCTOSPI_CCR.
7. Specify the instruction in OCTOSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
9. Specify the optional targeted address in OCTOSPI_AR.

If the address register (OCTOSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

Memory-mapped mode configuration

In Memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the Flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled.

Memory-mapped mode is entered by setting FMODE[1:0] = 11 in OCTOSPI_CR.

The programmed instruction and frame are sent when an AHB master accesses the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in Memory-mapped mode.

When the OCTOSPI is used in Memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in OCTOSPI_TCR for read operation.
2. Specify the frame format in OCTOSPI_CCR for read operation.
3. Specify the instruction in OCTOSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR for read operation.
5. Specify the frame timing in OCTOSPI_WTCR for write operation.
6. Specify the frame format in OCTOSPI_WCCR for write operation.
7. Specify the instruction in OCTOSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_WABR for read operation.

All the configuration operations must be completed before the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no further configuration is allowed.

OCTOSPI delayed data sampling when no DQS is used

By default, when no DQS is used, the OCTOSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in OCTOSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

OCTOSPI delayed data sampling when DQS is used

When external DQS is used as a sampling clock, it can be shifted in time to compensate the data propagation delay. This shift is performed by an external delay block located outside the OCTOSPI. The control of this feature depends on the device implementation (see the product reference manual for more details).

In certain configuration cases, this external delay block is implemented but is not useful, so it can be bypassed by setting DLYBYP in OCTOSPI_DCR1.

Sending the instruction only once (SIOO)

A Flash memory can provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. The user can take advantage of this type of features using SIOO in OCTOSPI_CCR.

SIOO is valid for Memory-mapped mode only. If this bit is set, the instruction is sent only for the first command following a write to OCTOSPI_CCR.

Subsequent command sequences skip the instruction phase, until there is a write to OCTOSPI_CCR. SIOO has no effect when IMODE[1:0] = 00 (no instruction).

SIOO mode is not supported when any of the communication regulation, NCS boundary or refresh features are used.

20.4.15 OCTOSPI HyperBus protocol configuration

Indirect mode configuration

When FMODE[1:0] = 00, the Indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the Indirect-read mode is selected where data can be read from the external device.

When the OCTOSPI 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 OCTOSPI_DLR.
2. Specify the targeted address in OCTOSPI_AR.
3. Make a write operation in OCTOSPI_IR and enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

In Indirect-read mode, the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In Indirect-write mode, the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

Automatic status-polling mode configuration

The Automatic status-polling mode is enabled setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and the data is retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in OCTOSPI_DLR, it is ignored and only 4 bytes are read. The periodicity is specified in OCTOSPI_PIR.

Once the status data has been retrieved, it can be internally processed to:

- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in OCTOSPI_PSMKR and can be ORed or ANDed with the value stored in OCTOSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The OCTOSPI can be automatically stopped if AMPS is set.

In any case, the latest retrieved value is available in OCTOSPI_DR.

When the OCTOSPI is used in Automatic status-polling mode, the frames are constructed in the following way:

1. Specify the input mask in OCTOSPI_PSMKR.
2. Specify the comparison value in OCTOSPI_PSMAR.
3. Specify the read period in OCTOSPI_PIR.
4. Specify a number of data bytes to read in OCTOSPI_DLR.
5. Specify the targeted address in OCTOSPI_AR.

The command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

Memory-mapped mode configuration

In Memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

The Memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AHB master accesses the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in Memory-mapped mode.

All the configuration operation must be completed prior to the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no configuration is allowed.

20.4.16 OCTOSPI error management

A error can be generated in the following cases:

- in Indirect or Automatic status-polling mode, when a wrong address has been programmed in OCTOSPI_AR (according to the device size defined by DEVSIZE[4:0]). This sets TEF and an interrupt is generated if enabled.
- in Indirect mode, if the address plus the data length exceed the device size. TEF is set as soon as the access is triggered.
- in Memory-mapped mode when an out-of-range access is done by an AHB master. This generates an AHB error as a response to the faulty AHB request.
- when the Memory-mapped mode is disabled. An access to the memory-mapped area generates an AHB error as a response to the faulty AHB request.

The OCTOSPI generates an AHB slave error in the following situations:

- Memory-mapped mode is disabled and an AHB read request occurs.
- Read or write address exceeds the size of the external memory.
- Abort is received while a read or write burst is ongoing.
- OCTOSPI is disabled while a read or write burst is ongoing.
- Write wrap burst is received.
- Write request is received while DQSE = 0 in OCTOSPI_WCCR, which means that the DQS output is disabled.
- Write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means the HSIZE is different from 4 bytes (only for Memory-mapped mode).
- Illegal wrap size when receiving read wrap burst with size different from 4 bytes (only for Memory-mapped mode).

20.4.17 OCTOSPI BUSY and ABORT

Once the OCTOSPI starts an operation with the external device, BUSY is automatically set in OCTOSPI_SR.

In Indirect mode, BUSY is reset once the OCTOSPI has completed the requested command sequence and the FIFO is empty.

In Automatic status-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 an abort.

Any operation can be aborted by setting ABORT in OCTOSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers.

Note: Some devices may misbehave if a write operation to a status register is aborted.

20.4.18 OCTOSPI reconfiguration or deactivation

Prior to any OCTOSPI reconfiguration, the software must ensure that all the transactions are completed:

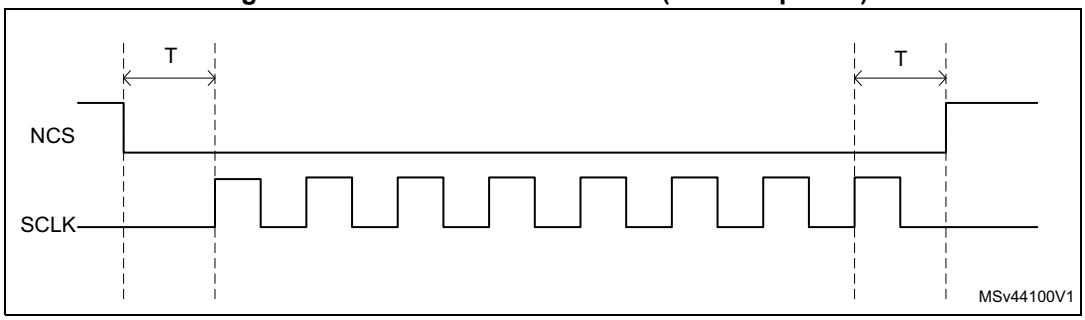
- After a Memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a Memory-mapped read, the software must perform a synchronization barrier then an abort.

20.4.19 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

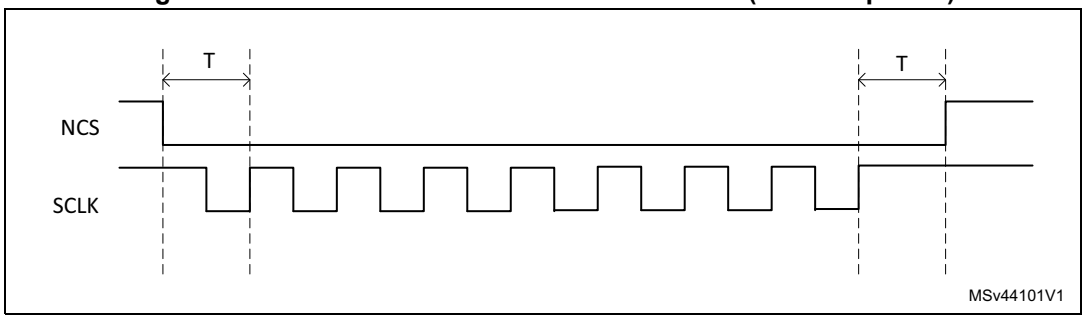
When CKMODE = 0 (Mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 87. NCS when CKMODE = 0 (T = CLK period)



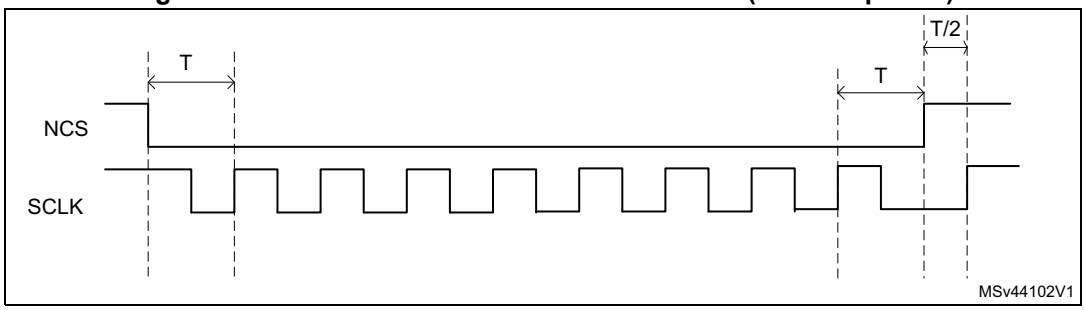
When CKMODE = 1 (Mode 3: CLK goes high when no operation is in progress) and when in SDR mode, NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 88. NCS when CKMODE = 1 in SDR mode (T = CLK period)



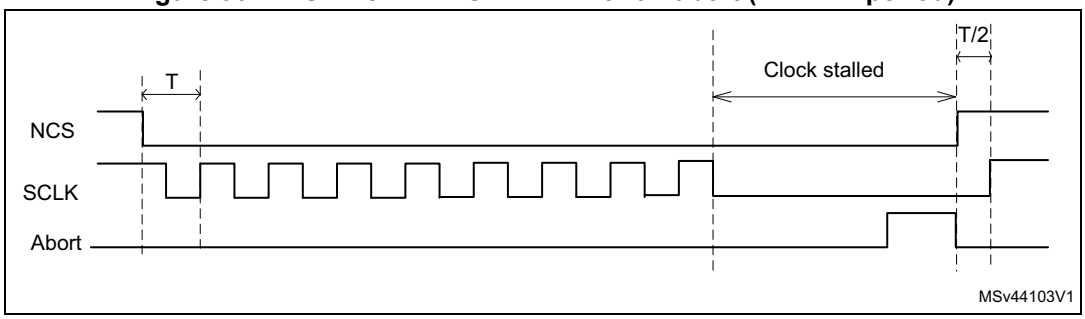
When the CKMODE = 1 (Mode 3) and DDTR = 1 (data DTR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge (see the figure below). Because the DTR operations must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Figure 89. NCS when CKMODE = 1 in DTR mode (T = CLK period)



When the FIFO stays full during a read operation, or if the FIFO stays empty during a write operation, the operation stalls and CLK stays low until the software 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 (see the figure below).

Figure 90. NCS when CKMODE = 1 with an abort (T = CLK period)



When not in dual-quad configuration (DMM = 0), only device A is accessed and thus the BNCS stays high. In dual-quad configuration, the BNCS behaves exactly the same as the ANCS. Thus, if there is a device B and if the application always stays in dual-quad

configuration, then the device B may use the ANCS and the pin outputting BNCS can be used for other functions.

20.5 Address alignment and data number

The following table summarizes the effect of the address alignment and programmed data number depending on the use case.

Table 153. Address alignment cases

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Single, dual, quad Flash or SRAM (DMM = 0)	IND(2) read	None	None	None	None
	MM(3) read				
	IND write				
	MM write				
Single, dual, quad Flash or SRAM (DMM = 1)	IND read	Even	ADDR[0] is set to 0.	Even	DLR[0] is set to 1.
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.	Even	DLR[0] is set to 1.
	MM write	Even	Slave error	Even	Last byte is lost.
Octal Flash in SDR mode	IND read	None	None	None	None
	MM read				
	IND write				
	MM write				
Octal Flash or RAM in DTR mode without RDS nor WDM(4)	IND read	Even	ADDR[0] is set to 0.	Even	DLR[0] is set to 1.
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.	Even	DLR[0] is set to 1.
	MM write	Even	Slave error	Even	Last byte is lost.
Octal Flash or RAM in DTR mode with RDS or WDM	IND read	Even	ADDR[0] is set to 0.	Even	DLR[0] is set to 1.
	MM read	None	None	None	None
	IND write				
	MM write				
HyperBus	IND read	Even	ADDR[0] is set to 0.	Even	DLR[0] is set to 1.
	MM read	None	None	None	None
	IND write				
	MM write				

1. To be respected by the software.
2. IND = Indirect mode.
3. MM = Memory-mapped mode
4. RDS = read data strobe, WDM = write data mask.

20.6 OCTOSPI 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 to provide more flexibility.

Table 154. OCTOSPI 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

20.7 OCTOSPI registers

20.7.1 OCTOSPI control register (OCTOSPI_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FMODE[1:0]		Res.	Res.	Res.	Res.	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
		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.	FTHRES[4:0]					MSEL	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
			rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **FMODE[1:0]**: Functional mode
This field defines the OCTOSPI functional mode of operation.
00: Indirect-write mode
01: Indirect-read mode
10: Automatic status-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[1:0] value. If FMODE[1:0] and FTHRES[4:0] are wrongly updated while DMAEN = 1, the DMA request signal automatically goes to inactive state.

Bits 27:24 Reserved, must be kept at reset value.

Bit 23 **PMM**: Polling match mode
This bit indicates which method must be used to determine a match during the Automatic status-polling mode.
0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.
1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.

Bit 22 **APMS**: Automatic status-polling mode stop
This bit determines if the Automatic status-polling mode is stopped after a match.
0: Automatic status-polling mode is stopped only by abort or by disabling the OCTOSPI.
1: Automatic status-polling mode stops as soon as there is a match.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: Timeout interrupt enable
This bit enables the timeout interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **SMIE**: Status match interrupt enable
This bit enables the status match interrupt.
0: Interrupt disabled
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 disabled
1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level

This field defines, in Indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in OCTOSPI_SR, to be set.

00000: FTF is set if there are one or more free bytes available to be written to in the FIFO in Indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in Indirect-read mode.

00001: FTF is set if there are two or more free bytes available to be written to in the FIFO in Indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in Indirect-read mode.

...

11111: FTF is set if there are 32 free bytes available to be written to in the FIFO in Indirect-write mode, or if there are 32 valid bytes can be read from the FIFO in Indirect-read mode.

Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[4:0] value.

Bit 7 **MSEL**: Flash select

This bit selects the Flash memory to be addressed in Single-, Dual-, Quad-SPI mode in single-memory configuration (when DMM = 0).

0: FLASH 1 selected (data exchanged over IO[3:0])

1: FLASH 2 selected (data exchanged over IO[7:4])

This bit is ignored when DMM = 1 or when Octal-SPI mode is selected.

Bit 6 **DMM**: Dual-memory configuration

This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity

0: Dual-quad configuration disabled

1: Dual-quad configuration enabled

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when the Memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.

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 the Memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

Bit 2 **DMAEN**: DMA enable

In Indirect mode, the DMA can be used to input or output data via OCTOSPI_DR. DMA transfers are initiated when FTF is set.

0: DMA disabled for Indirect mode

1: DMA enabled for Indirect mode

Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.

Bit 1 **ABORT**: Abort request

This bit aborts the on-going command sequence. It is automatically reset once the abort is completed. This bit stops the current transfer.

0: No abort requested

1: Abort requested

Note: This bit is always read as 0.

Bit 0 **EN**: Enable

This bit enables the OCTOSPI.

0: OCTOSPI disabled

1: OCTOSPI enabled

Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation.
In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.

20.7.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1)

Address offset: 0x0008

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	MTYP[2:0]			Res.	Res.	Res.	DEVSIZE[4:0]				
					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.	CSHT[2:0]			Res.	Res.	Res.	Res.	DLY BYP	Res.	FRCK	CKMODE
					rw	rw	rw					rw		rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **MTYP[2:0]**: Memory type

This bit indicates the type of memory to be supported.

000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. Regular-command protocol in Single-, Dual-, Quad- and Octal-SPI modes.

Note: In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.

001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in Single-, Dual-, Quad- and Octal-SPI modes.

010: Standard mode

011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in Single-, Dual-, Quad- and Octal-SPI modes with dedicated address mapping.

100: HyperBus memory mode, the protocol follows the HyperBus specification. 8-data-bit DTR mode must be selected.

101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be non-cacheable, or Indirect read/write modes must be used.

Others: Reserved

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

Bits 20:16 **DEVSIZE[4:0]**: Device size

This field defines the size of the external device using the following formula:
Number of bytes in device = $2^{[DEVSIZE+1]}$.
DEVSIZE+1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in Indirect mode, but the addressable space in Memory-mapped mode is limited to 256 Mbytes.
In Regular-command protocol, if DMM = 1, DEVSIZE[4:0] indicates the total capacity of the two devices together.

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 where the chip-select (NCS) must remain high between commands issued to the external device.
000: NCS stays high for at least 1 cycle between external device commands.
001: NCS stays high for at least 2 cycles between external device commands.
...
111: NCS stays high for at least 8 cycles between external device commands.

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

Bit 3 **DLYBYP**: Delay block bypass

0: The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the OCTOSPI peripheral).
1: The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock.
0: CLK is not free running.
1: CLK is free running (always provided).

Bit 0 **CKMODE**: Mode 0/Mode 3

This bit indicates the level taken by the CLK 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.

20.7.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2)

Address offset: 0x000C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 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.	WRAPSIZE[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.	PRESCALER[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 18:16 **WRAPSIZE[2:0]**: Wrap size

This field indicates the wrap size to which the memory is configured. For memories which have a separate command for wrapped instructions, this field indicates the wrap-size associated with the command held in the OCTOSPI1_WPIR register.

000: Wrapped reads are not supported by the memory.

001: Reserved

010: External memory supports wrap size of 16 bytes.

011: External memory supports wrap size of 32 bytes.

100: External memory supports wrap size of 64 bytes.

101: External memory supports wrap size of 128 bytes.

110-111: Reserved

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

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This field defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as OCTOSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the OCTOSPI a kernel clock that has 50% duty-cycle.

1: $F_{CLK} = F_{KERNEL}/2$

2: $F_{CLK} = F_{KERNEL}/3$

...

255: $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK duty cycle is not 50 %. The clock signal remains low one cycle longer than it stays high.

20.7.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3)

Address offset: 0x0010

Reset value: 0x0000 0000

This register can be modified only when BUSY = 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.	CSBOUND[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.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]**: NCS boundary

This field enables the transaction boundary feature. When active, a minimum value of 3 is recommended.
The NCS is released on each boundary of $2^{CSBOUND}$ bytes.
0: NCS boundary disabled
others: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:0 Reserved, must be kept at reset value.

20.7.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4)

Address offset: 0x0014

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **REFRESH[31:0]**: Refresh rate

This field enables the refresh rate feature.
The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads.

Note: These two values can be extended with few clock cycles when refresh occurs during a byte transmission in Single-, Dual- or Quad-SPI mode, because the byte transmission must be completed.

0: Refresh disabled
others: Maximum communication length is set to REFRESH + 1 clock cycles.

20.7.6 OCTOSPI status register (OCTOSPI_SR)

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.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
		r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This field gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full.

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 ongoing. It is cleared automatically when the operation with the external device 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 status-polling mode when the unmasked received data matches the corresponding bits in the match register (OCTOSPI_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 has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.

It is cleared automatically as soon as the threshold condition is no longer true.

In Automatic status-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 has been transferred or in any mode when the transfer has been 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.

20.7.7 OCTOSPI flag clear register (OCTOSPI_FCR)

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
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 OCTOSPI_SR register.

Bit 3 **CSMF**: Clear status match flag
Writing 1 clears the SMF flag in the OCTOSPI_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 OCTOSPI_SR register.

Bit 0 **CTEF**: Clear transfer error flag
Writing 1 clears the TEF flag in the OCTOSPI_SR register.

20.7.8 OCTOSPI data length register (OCTOSPI_DLR)

Address offset: 0x0040

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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 Automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for Automatic status-polling mode.
All 1’s in Indirect mode means undefined length, where OCTOSPI continues until the end of the memory, as defined by DEVSIZE.
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 the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.
DL[0] is stuck at 1 in dual-memory configuration (DMM = 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 in Memory-mapped mode.

20.7.9 OCTOSPI address register (OCTOSPI_AR)

Address offset: 0x0048

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 sent to the external device. In HyperBus protocol, this field must be even as this protocol is 16-bit word oriented. In dual-memory configuration, AR[0] is forced to 1.
Writes to this field are ignored when BUSY = 1 or when FMODE = 11 (Memory-mapped mode).

20.7.10 OCTOSPI data register (OCTOSPI_DR)

Address offset: 0x0050

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 external device 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 that was received from the external device. 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 status-polling mode, this register contains the last data read from the external device (without masking).
Word, half-word, and byte accesses to this register are supported. In Indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes.
Similarly, in Indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in Indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

20.7.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR)

Address offset: 0x0080

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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 Automatic status-polling mode
For bit n:
0: Bit n of the data received in Automatic status-polling mode is masked and its value is not considered in the matching logic.
1: Bit n of the data received in Automatic status-polling mode is unmasked and its value is considered in the matching logic.

20.7.12 OCTOSPI polling status match register (OCTOSPI_PSMAR)

Address offset: 0x0088

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

20.7.13 OCTOSPI polling interval register (OCTOSPI_PIR)

Address offset: 0x0090

Reset value: 0x0000 0000

This register can be modified only when BUSY = 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
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 cycle between a read during the Automatic status-polling phases

20.7.14 OCTOSPI communication configuration register (OCTOSPI_CCR)

Address offset: 0x0100

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIOO	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[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.	ADSIZE[1:0]		AD DTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 **SIOO**: Send instruction only once mode
This bit has no effect when IMODE = 00 (see *Sending the instruction only once (SIOO)*).
0: Send instruction on every transaction
1: Send instruction only for the first command

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for data phase
1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This field defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
101-111: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate bytes size
This bit defines alternate bytes size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate bytes phase.
0: DTR mode disabled for alternate bytes phase
1: DTR mode enabled for alternate bytes phase
This field can be written only when BUSY = 0.

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This field defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
101-111: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This field defines address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This field defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
101-111: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bit defines instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This field defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
101-111: Reserved

20.7.15 OCTOSPI timing configuration register (OCTOSPI_TCR)

Address offset: 0x0108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift

By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle

0: No delay hold
1: 1/4 cycle hold

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This field defines the duration of the dummy phase.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).
It is recommended to have at least six dummy cycles when using memories with DQS activated.

20.7.16 OCTOSPI instruction register (OCTOSPI_IR)

Address offset: 0x0110

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction

Instruction to be sent to the external SPI device

20.7.17 OCTOSPI alternate bytes register (OCTOSPI_ABR)

Address offset: 0x0120

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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 sent to the external SPI device right after the address.

20.7.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR)

Address offset: 0x00130

Reset value: 0x0000 0000

This register can be modified only when BUSY = 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
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 OCTOSPI prefetches the subsequent bytes and hold them in the FIFO.

This field indicates how many CLK cycles the OCTOSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

20.7.19 OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR)

Address offset: 0x0140

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		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.	ADSIZE[1:0]		AD DTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for data phase
1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This field defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
101-111: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate bytes size
This bit defines alternate bytes size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate bytes phase.
0: DTR mode disabled for alternate bytes phase
1: DTR mode enabled for alternate bytes phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This field defines the alternate byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
101-111: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This field defines address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This field defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
101-111: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This field defines instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This field defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
101-111: Reserved

20.7.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR)

Address offset: 0x0148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift

By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.

This bit allows the data to be sampled later in order to consider the external signal delays.

0: No shift

1: 1/2 cycle shift

The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR = 1).

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle

Add a quarter cycle delay on the outputs in DTR communication to match hold requirement.

0: No quarter cycle delay

1: Quarter cycle delay inserted

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This field defines the duration of the dummy phase.

In both SDR and DTR modes, it specifies a number of CLK cycles (0-31). It is recommended to have at least 5 dummy cycles when using memories with DQS activated.

20.7.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR)

Address offset: 0x0150

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31: 0]**: Instruction

Instruction to be sent to the external SPI device

20.7.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR)

Address offset: 0x0160

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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 sent to the external SPI device right after the address

20.7.23 OCTOSPI write communication configuration register (OCTOSPI_WCCR)

Address offset: 0x0180

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		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.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for data phase
1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This field defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
101-111: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate bytes size
This field defines alternate bytes size:
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate-bytes phase.
0: DTR mode disabled for alternate-bytes phase
1: DTR mode enabled for alternate-bytes phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This field defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
101-111: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This field defines address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This field defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
101-111: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bit defines instruction size:
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This field defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
101-111: Reserved

20.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR)

Address offset: 0x0188

Reset value: 0x0000 0000

This register can be modified only when BUSY = 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.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This field defines the duration of the dummy phase.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31). It is recommended to have at least 5 dummy cycles when using memories with DQS activated.

20.7.25 OCTOSPI write instruction register (OCTOSPI_WIR)

Address offset: 0x0190

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

20.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR)

Address offset: 0x01A0

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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 sent to the external SPI device right after the address

20.7.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR)

Address offset: 0x0200

Reset value: 0x0000 0000

This register can be modified only when BUSY = 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.	TRWR[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
TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw

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

Bits 23:16 **TRWR[7:0]:** Read write recovery time

Device read write recovery time expressed in number of communication clock cycles

Bits 15:8 **TACC[7: 0]**: Access time

Device access time expressed in number of communication clock cycles

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

Bit 1 **WZL:** Write zero latency

This bit enables zero latency on write operations.

0: Latency on write accesses

1: No latency on write accesses

Bit 0 **LM:** Latency mode

This bit selects the Latency mode.

0: Variable initial latency

1: Fixed latency

20.7.28 OCTOSPI register map

Table 155. OCTOSPI 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
0x0000	OCTOSPI_CR	Res.	Res.	FMODE[1:0]		Res.	Res.	Res.	Res.	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES[4:0]					MSEL	DMM	Res.	Res.	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
0x0004	Reserved	Reserved																															
0x0008	OCTOSPI_DCR1	Res.	Res.	Res.	Res.	Res.	MTYP [2:0]			Res.	Res.	Res.	DEVSIZE[4:0]					Res.	Res.	Res.	Res.	Res.	CSHT [2:0]			Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
	Reset value						0	0	0				0	0	0	0	0						0	0	0					0		0	0
0x000C	OCTOSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
	Reset value														0	0	0									0	0	0	0	0	0	0	0
0x0010	OCTOSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0	0	0	0																
0x0014	OCTOSPI_DCR4	REFRESH[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
0x0018-0x001C	Reserved	Reserved																															
0x0020	OCTOSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
	Reset value																			0	0	0	0	0	0			0	0	0	0	0	0
0x0024	OCTOSPI_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
0x0028-0x003C	Reserved	Reserved																															
0x0040	OCTOSPI_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

Table 155. OCTOSPI 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
0x0044	Reserved	Reserved																															
0x0048	OCTOSPI_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
0x004C	Reserved	Reserved																															
0x0050	OCTOSPI_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
0x0054-0x007C	Reserved	Reserved																															
0x0080	OCTOSPI_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
0x0084	Reserved	Reserved																															
0x0088	OCTOSPI_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
0x008C	Reserved	Reserved																															
0x0090	OCTOSPI_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
0x0094-0x00FC	Reserved	Reserved																															
0x0100	OCTOSPI_CCR	SIOO	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE [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
0x0104	Reserved	Reserved																															
0x0108	OCTOSPI_TCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x010C	Reserved	Reserved																															
0x0110	OCTOSPI_IR	INSTRUCTION[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
0x0114-0x011C	Reserved	Reserved																															
0x0120	OCTOSPI_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
0x0124-0x012C	Reserved	Reserved																															
0x0130	OCTOSPI_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
0x0134-0x013C	Reserved	Reserved																															

Table 155. OCTOSPI 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
0x0140	OCTOSPI_ WPCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [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
0x0144	Reserved	Reserved																															
0x0148	OCTOSPI_ WPTCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x014C	Reserved	Reserved																															
0x0150	OCTOSPI_WPIR	INSTRUCTION[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
0x0154- 0x015C	Reserved	Reserved																															
0x0160	OCTOSPI_ WPABR	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
0x0164- 0x017C	Reserved	Reserved																															
0x0180	OCTOSPI_WCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [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
0x0184	Reserved	Reserved																															
0x0188	OCTOSPI_WTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value																												0	0	0	0	0
0x018C	Reserved	Reserved																															
0x0190	OCTOSPI_WIR	INSTRUCTION[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
0x0194- 0x019C	Reserved	Reserved																															
0x01A0	OCTOSPI_WABR	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
0x01A4- 0x01FC	Reserved	Reserved																															
0x0200	OCTOSPI_HLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]								TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0

Refer to *Section 2.3* for the register boundary addresses.

21 Analog-to-digital converters (ADC)

21.1 Introduction

This section describes the implementation of up to 2 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).

Each ADC consists of a 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 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 2 ADCs which can operate in dual mode:
    ADC1 is connected to 16 external channels + 3 internal channels
    ADC2 is connected to 16 external 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
  - 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
  - Data can be routed to DFSDM for post processing
  - 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
- 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 11 slow channels from GPIO pads
- In addition, there are several internal dedicated channels
  - The internal reference voltage (V_{REFINT}), connected to ADC1
  - The internal temperature sensor (V_{TS}), connected to ADC1
  - The V_{BAT} monitoring channel ($V_{BAT}/3$), connected to ADC1
  - DAC1 and DAC2 internal channels, connected to ADC2
- 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 and 2
- 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
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

Figure 91 shows the block diagram of one ADC.

21.3 ADC implementation

Table 156. ADC features

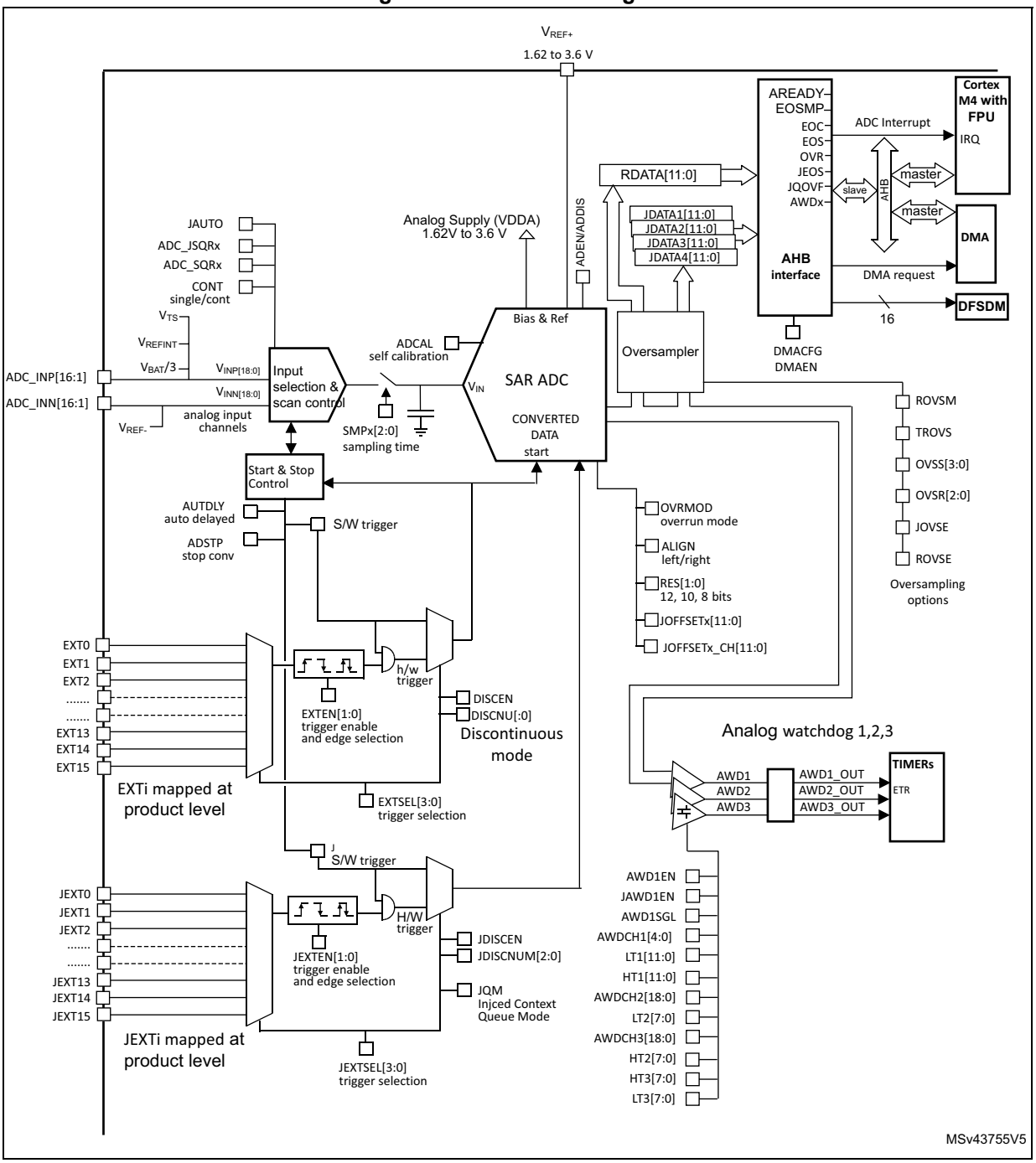
ADC modes/features	ADC1	ADC2
Dual mode	X	X
DFSDM interface	X	X
SMPPLUS control	-	-

21.4 ADC functional description

21.4.1 ADC block diagram

Figure 91 shows the ADC block diagram and *Table 158* gives the ADC pin description.

Figure 91. ADC block diagram



21.4.2 ADC pins and internal signals

Table 157. ADC internal input/output signals

Internal signal name	Signal type	Description
EXT[15:0]	Inputs	Up to 16 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.
JEXT[15:0]	Inputs	Up to 16 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)
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 158. 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].
ADCx_INNi	Negative external analog input signals	Up to 16 analog input channels (x = ADC number = 1 or 2) Refer to *Section 21.4.4: ADC1/2 connectivity* for details.
ADCx_INPi	Positive external analog input signals	Up to 10 analog input channels (x = ADC number = 1 or 2) Refer to *Section 21.4.4: ADC1/2 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 92: ADC clock scheme*):

1. The ADC clock can be a specific clock source, derived from the following clock sources:
   – The system clock
   – PLLSAI1 (single ADC implementation)

   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]).

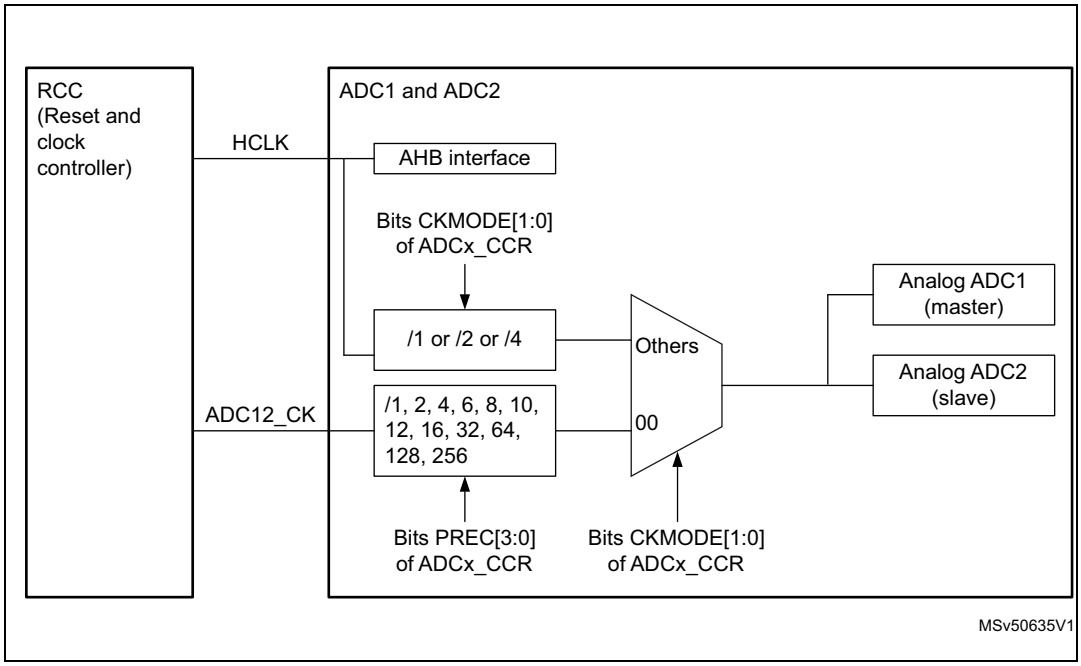
   To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be different from “00”.

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 92. ADC clock scheme



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:

- $F_{HCLK} \geq F_{ADC} / 4$ if the resolution of all channels are 12-bit or 10-bit
- $F_{HCLK} \geq F_{ADC} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolution)
- $F_{HCLK} \geq F_{ADC} / 2$ if there are some channels with resolutions equal to 6-bit

21.4.4 ADC1/2 connectivity

ADC1 and ADC2 are tightly coupled and share some external channels as described in the below figures.

Figure 93. ADC1 connectivity

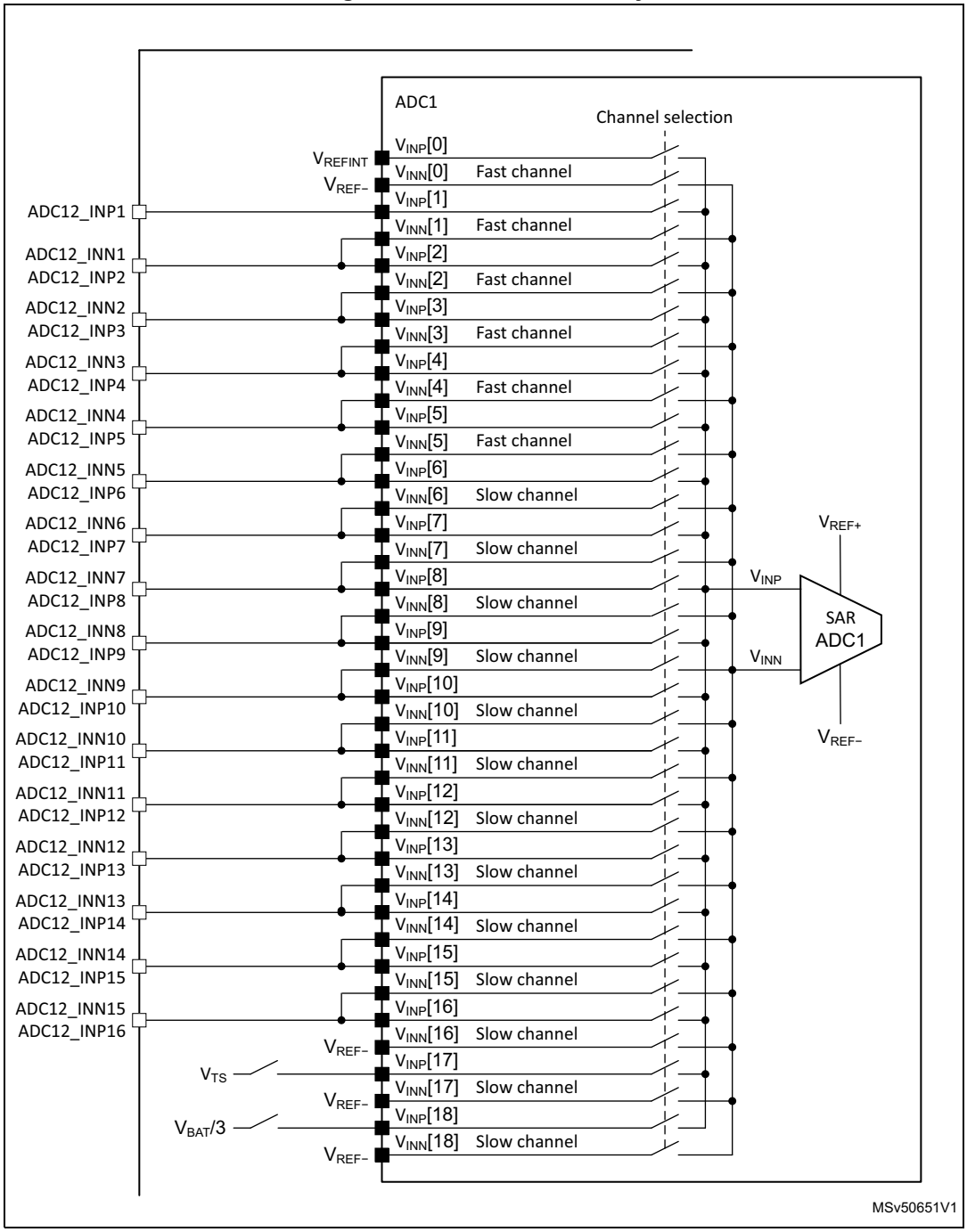
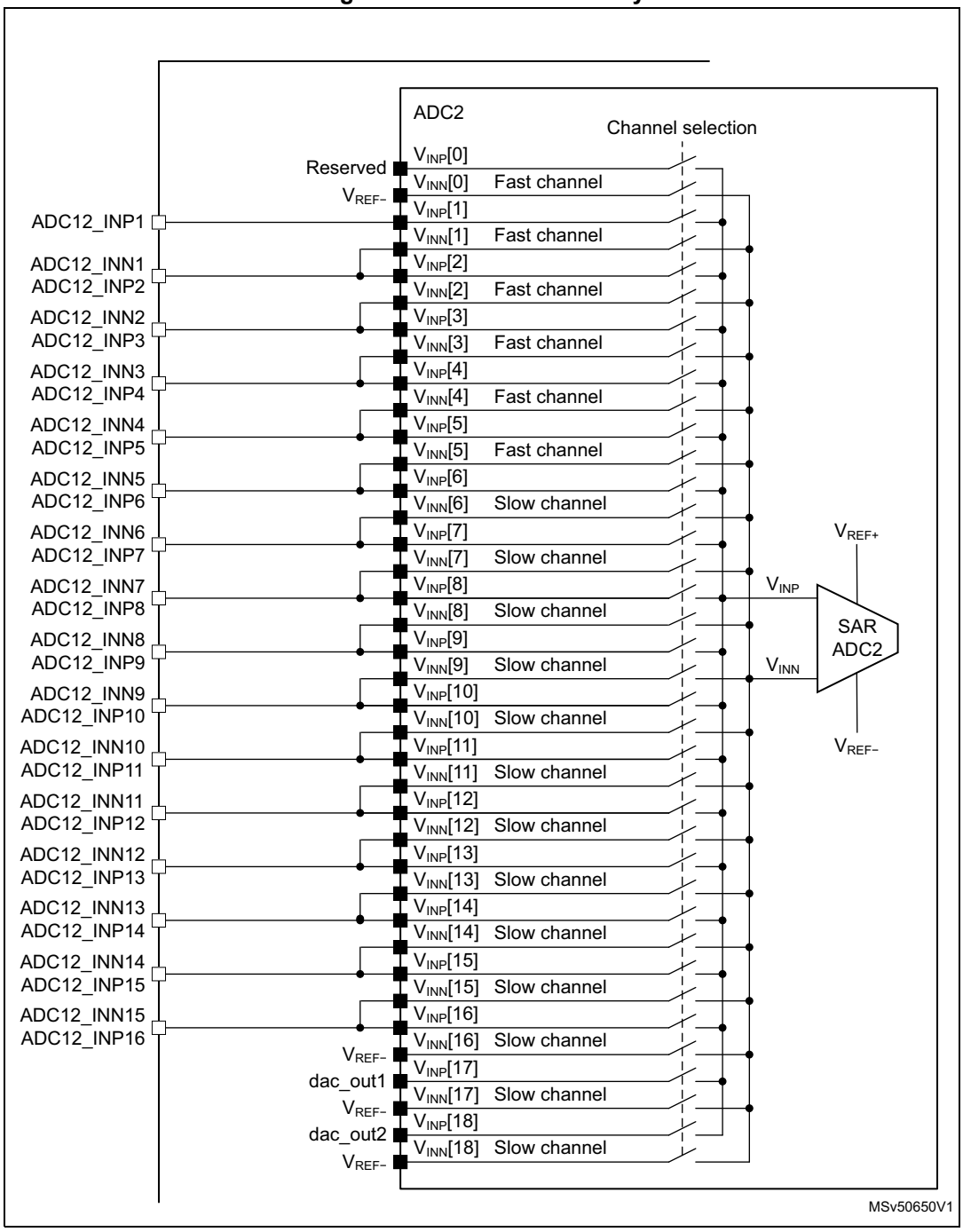


Figure 94. ADC2 connectivity



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 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: this can make the channel on the other ADC unusable. Only exception is interleaved mode for ADC master and the slave.

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.

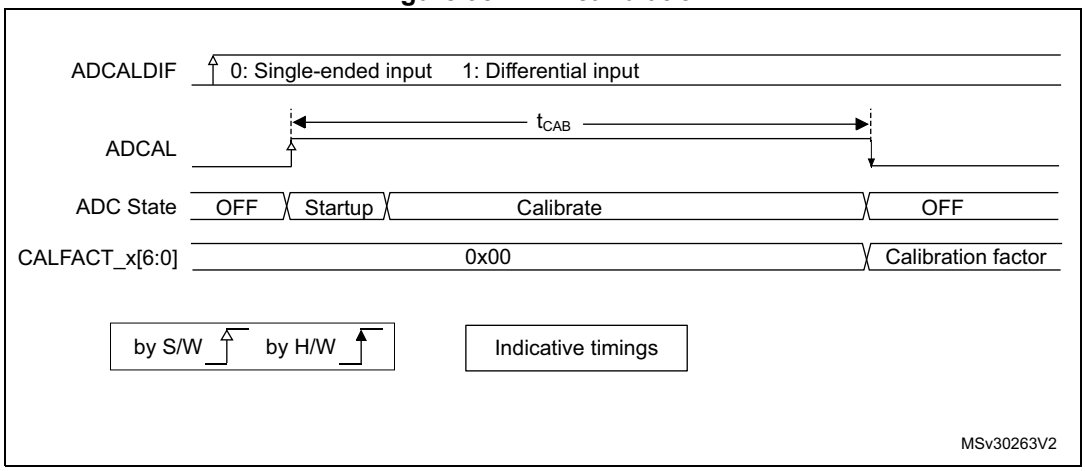
The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in Standby or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, 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.

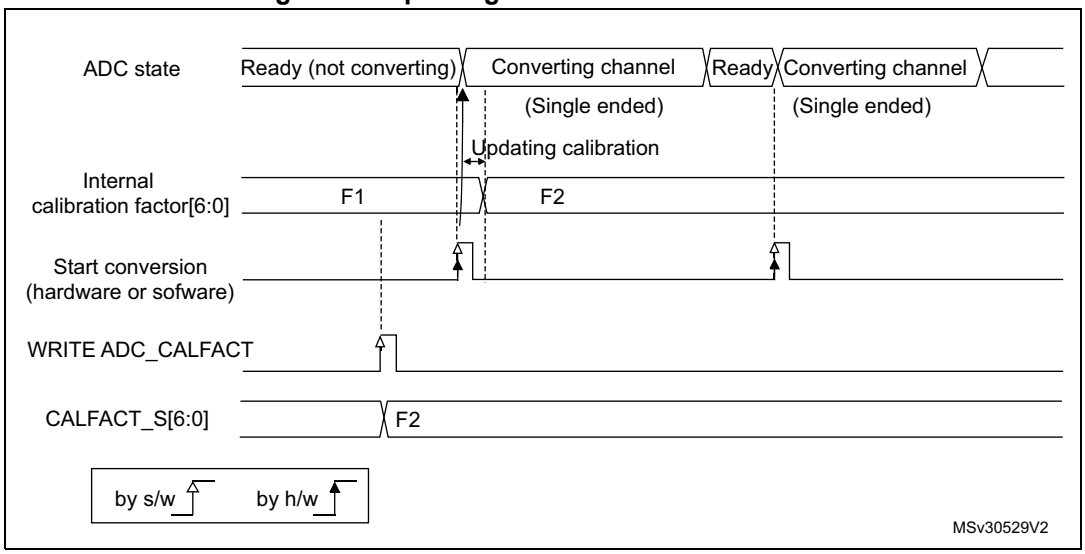
Figure 95. ADC calibration



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 96. Updating the ADC calibration factor

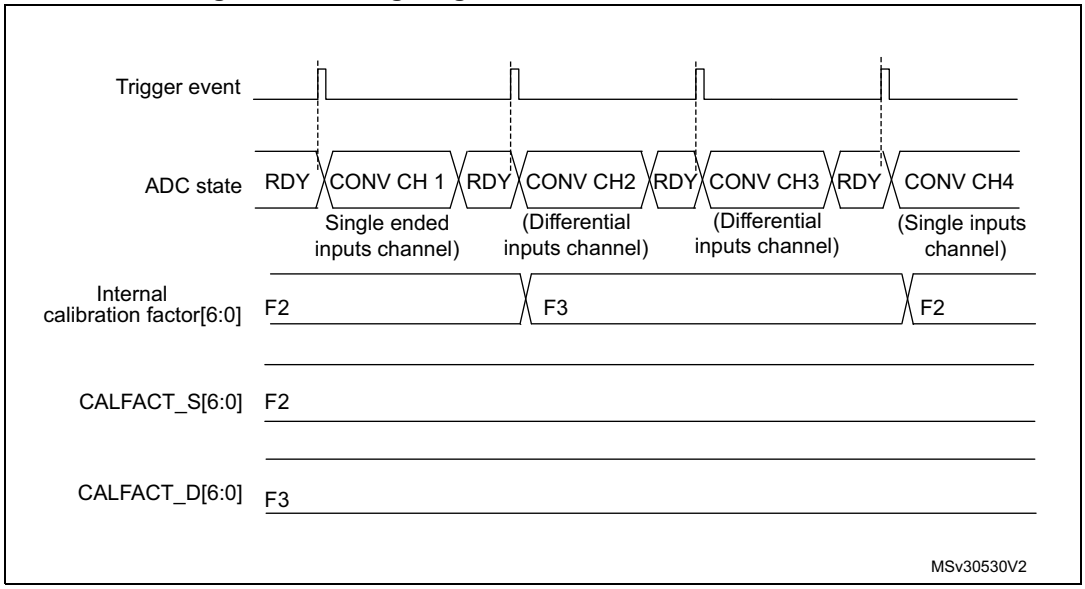


Converting single-ended and differential analog inputs with a single ADC

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 97. 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 98*. 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

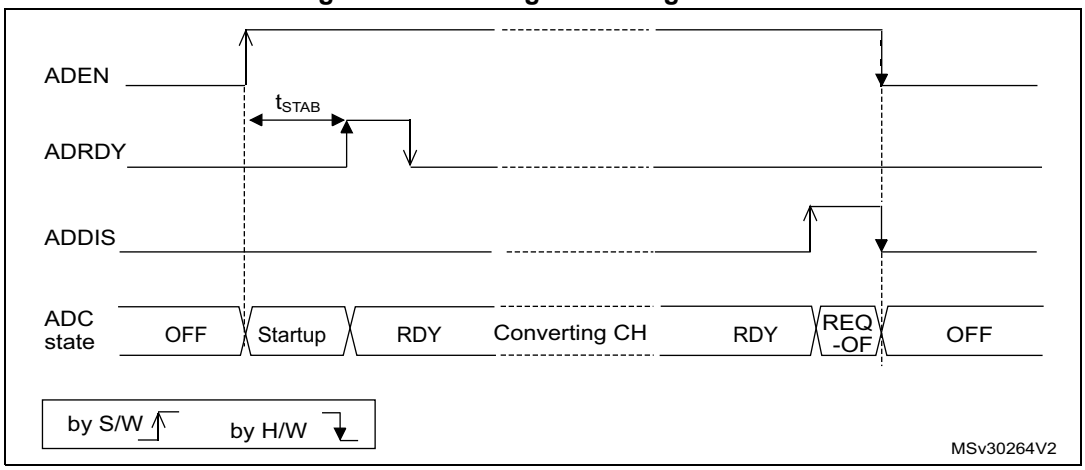
1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set ADEN=1.
3. Wait until ADRDY=1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE=1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

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 98. 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_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).

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 20 multiplexed channels per ADC:

- Up to 11 slow analog inputs coming from GPIO pads (ADCx_INP/INN[6:16])
  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_INP0/INN0.
  - The internal temperature sensor (V_{TS}) is connected to ADC1_INP17/INN17.
  - The V_{BAT} monitoring channel ($V_{BAT}/3$) is connected to ADC1_INP18/INN18.
  - The DAC1 internal channel 1 is connected to ADC2_INP/INN17.
  - The DAC1 internal channel 2 is connected to ADC2_INP/INN18.

Note: To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, CH17SEL or CH18SEL 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_CLK} = 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.

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.

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.

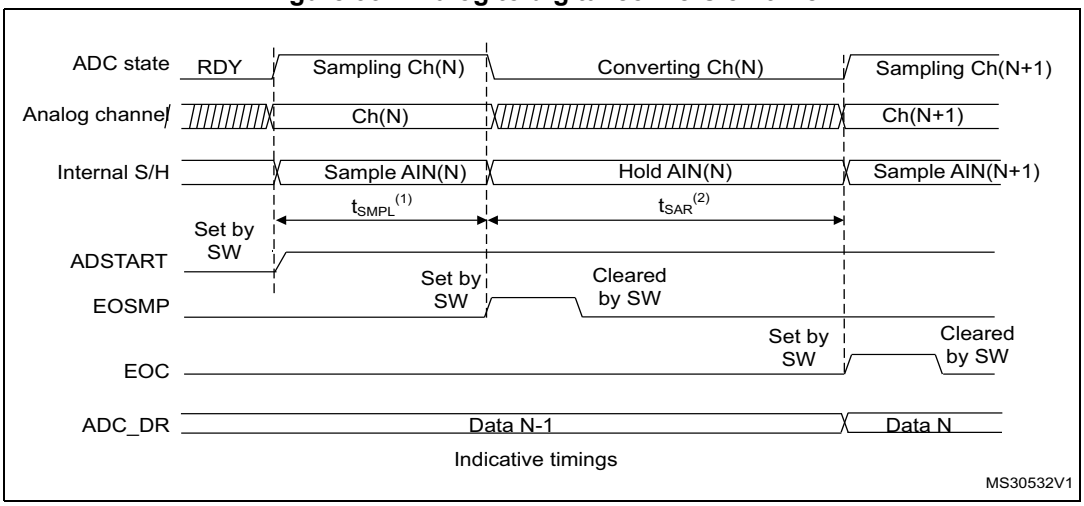
21.4.16 ADC timing

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:

$T_{CONV} = T_{SMPL} + T_{SAR} = [2.5\ |_{min} + 12.5\ |_{12bit}] \times T_{ADC\_CLK}$

$T_{CONV} = T_{SMPL} + T_{SAR} = 83.33\ ns\ |_{min} + 416.67\ ns\ |_{12bit} = 500.0\ ns$ (for $F_{ADC\_CLK} = 30$ MHz)

Figure 99. 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 100. Stopping ongoing regular conversions

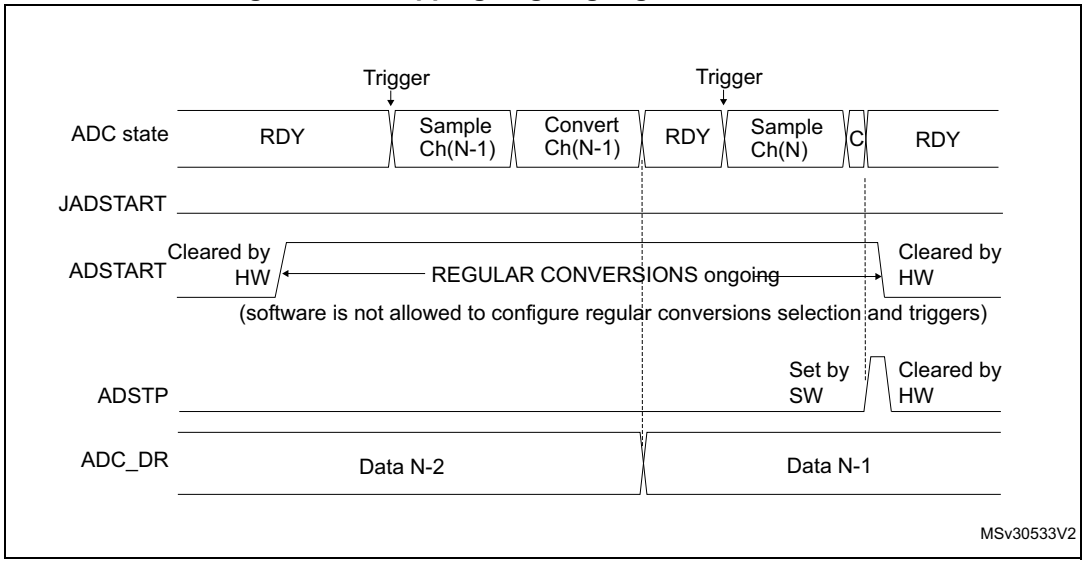
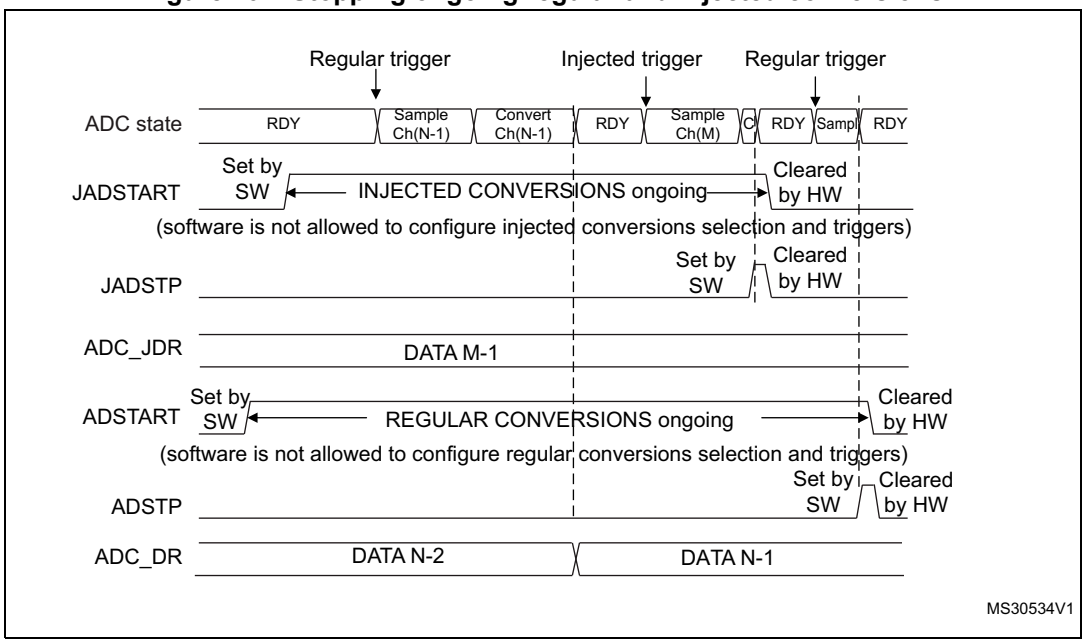


Figure 101. Stopping ongoing regular and injected conversions



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 159 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 159. 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 160. 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 16 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 710*

Each ADC master shares the same input triggers with its ADC slave as described in *Figure 102*.

Figure 102. Triggers sharing between ADC master and ADC slave

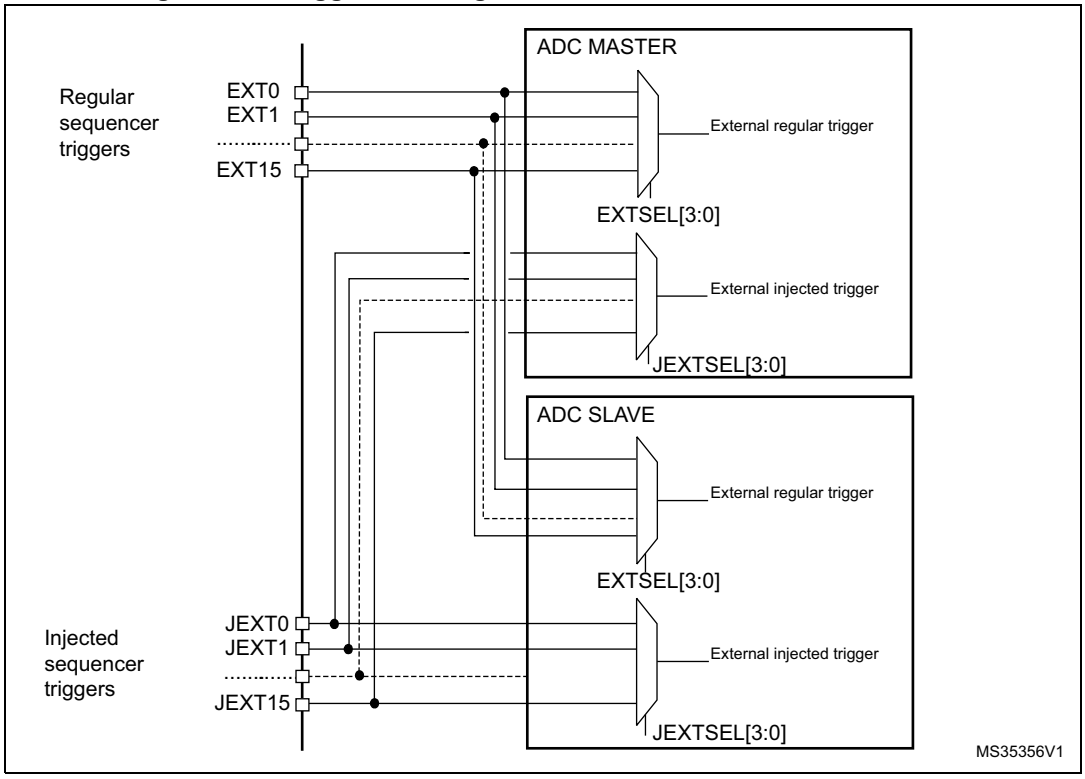


Table 161 to *Table 162* give all the possible external triggers of the three ADCs for regular and injected conversions.

Table 161. ADC1/2 - External triggers for regular channels

Name	Source	Type	EXTSEL[3:0]
EXT0	TIM1_CH1	Internal signal from on-chip timers	0000
EXT1	TIM1_CH2	Internal signal from on-chip timers	0001
EXT2	TIM1_CH3	Internal signal from on-chip timers	0010
EXT3	TIM2_CH2	Internal signal from on-chip timers	0011
EXT4	TIM3_TRGO	Internal signal from on-chip timers	0100
EXT5	TIM4_CH4	Internal signal from on-chip timers	0101
EXT6	EXTI line 11	External pin	0110
EXT7	TIM8_TRGO	Internal signal from on-chip timers	0111
EXT8	TIM8_TRGO2	Internal signal from on-chip timers	1000
EXT9	TIM1_TRGO	Internal signal from on-chip timers	1001

Table 161. ADC1/2 - External triggers for regular channels (continued)

Name	Source	Type	EXTSEL[3:0]
EXT10	TIM1_TRGO2	Internal signal from on-chip timers	1010
EXT11	TIM2_TRGO	Internal signal from on-chip timers	1011
EXT12	TIM4_TRGO	Internal signal from on-chip timers	1100
EXT13	TIM6_TRGO	Internal signal from on-chip timers	1101
EXT14	TIM15_TRGO	Internal signal from on-chip timers	1110
EXT15	TIM3_CH4	Internal signal from on-chip timers	1111

Table 162. ADC1/2 - External trigger for injected channels

Name	Source	Type	JEXTSEL[3:0]
JEXT0	TIM1_TRGO	Internal signal from on-chip timers	0000
JEXT1	TIM1_CH4	Internal signal from on-chip timers	0001
JEXT2	TIM2_TRGO	Internal signal from on-chip timers	0010
JEXT3	TIM2_CH1	Internal signal from on-chip timers	0011
JEXT4	TIM3_CH4	Internal signal from on-chip timers	0100
JEXT5	TIM4_TRGO	Internal signal from on-chip timers	0101
JEXT6	EXTI line 15	External pin	0110
JEXT7	TIM8_CH4	Internal signal from on-chip timers	0111
JEXT8	TIM1_TRGO2	Internal signal from on-chip timers	1000
JEXT9	TIM8_TRGO	Internal signal from on-chip timers	1001
JEXT10	TIM8_TRGO2	Internal signal from on-chip timers	1010
JEXT11	TIM3_CH3	Internal signal from on-chip timers	1011
JEXT12	TIM3_TRGO	Internal signal from on-chip timers	1100
JEXT13	TIM3_CH1	Internal signal from on-chip timers	1101
JEXT14	TIM6_TRGO	Internal signal from on-chip timers	1110
JEXT15	TIM15_TRGO	Internal signal from on-chip timers	1111

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 103* 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).

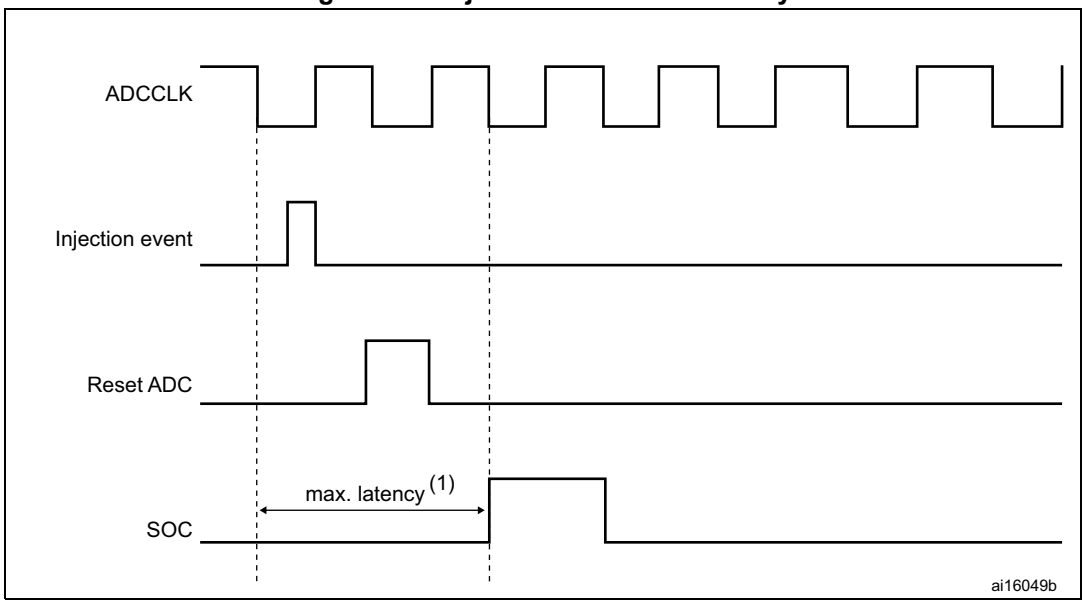
In this mode, external trigger on injected channels must be disabled.

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 103. 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 ≤ 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.

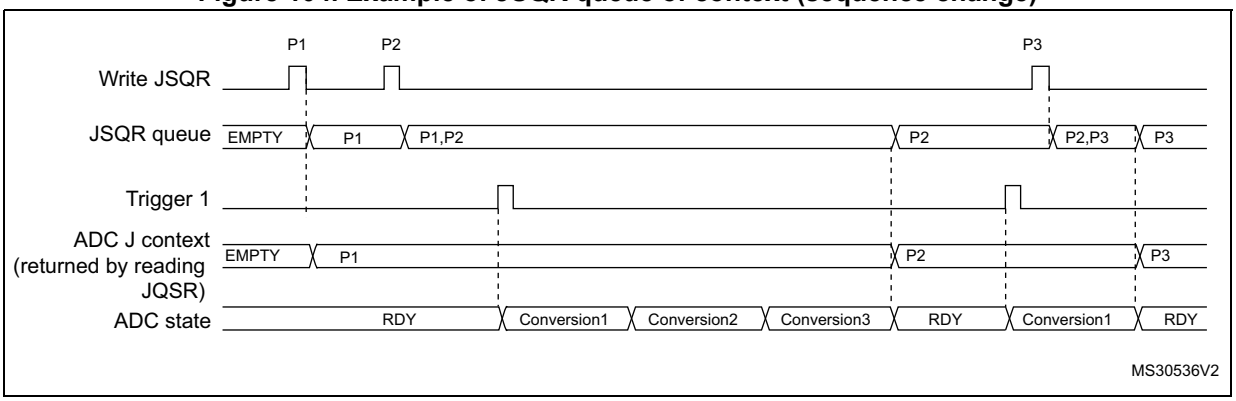
Note: When configured in discontinuous mode (bit JDISCEN=1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

- *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

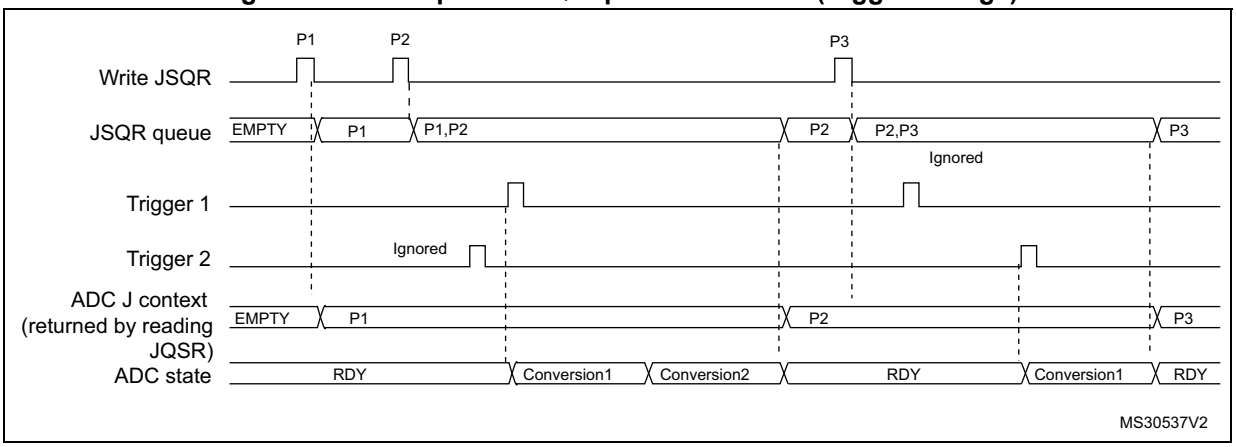
The *Figure 104* and *Figure 105* show the behavior of the context Queue when changing the sequence or the triggers.

Figure 104. 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 105. Example of JSQR queue of context (trigger change)

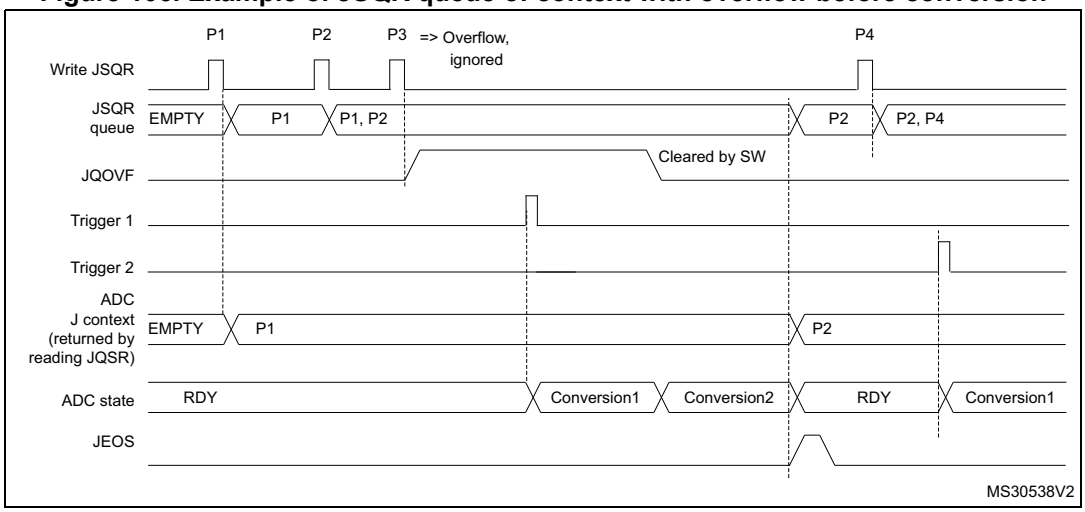


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

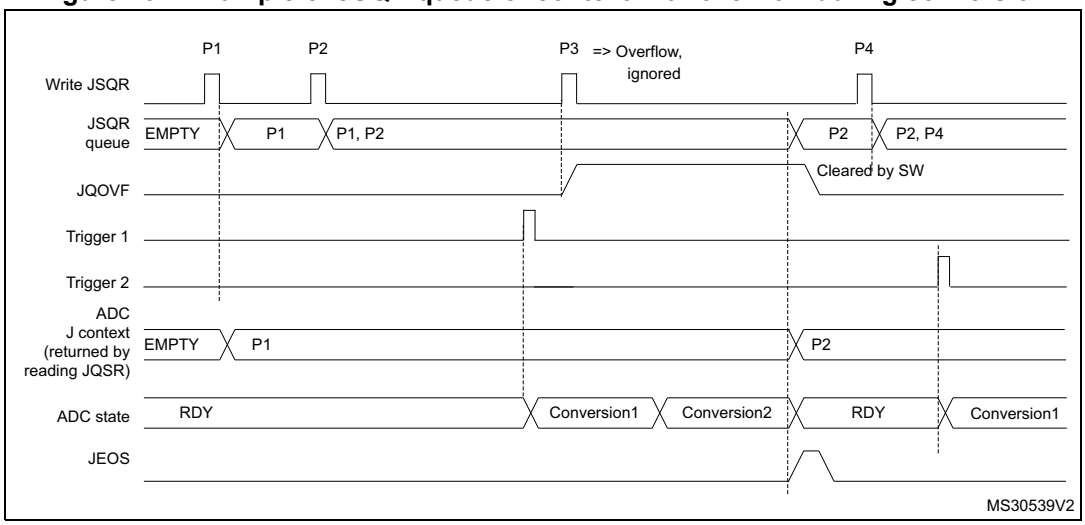
The *Figure 106* and *Figure 107* show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 106. 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 107. 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

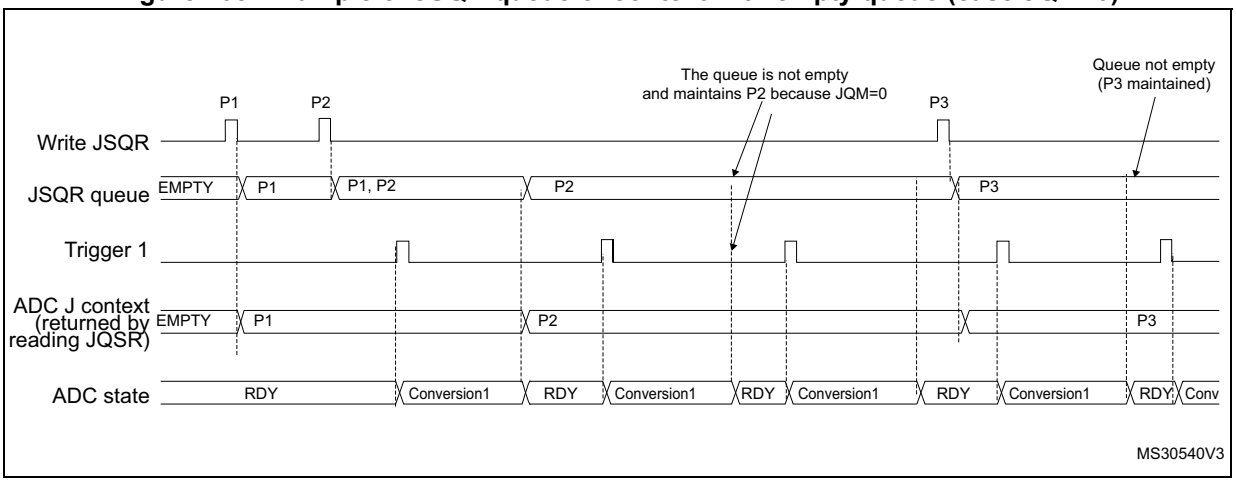
It is recommended to manage the queue overflows as described below:

- 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 108 and *Figure 109* show the behavior of the context Queue when the Queue becomes empty in both cases JQM=0 or 1.

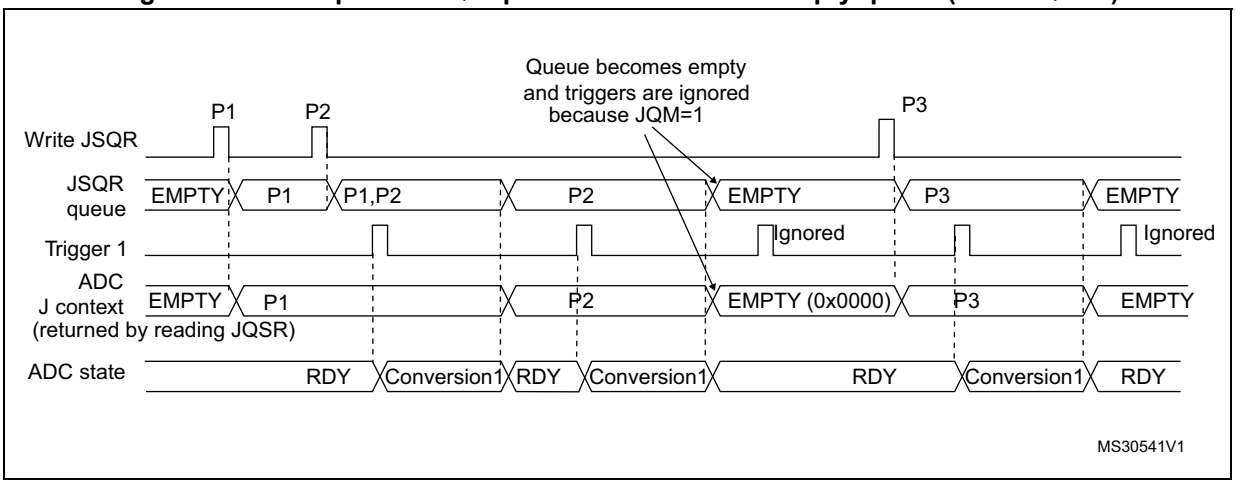
Figure 108. Example of JSQR queue of context with empty queue (case JQM=0)



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 109. Example of JSQR queue of context with empty queue (case JQM=1)

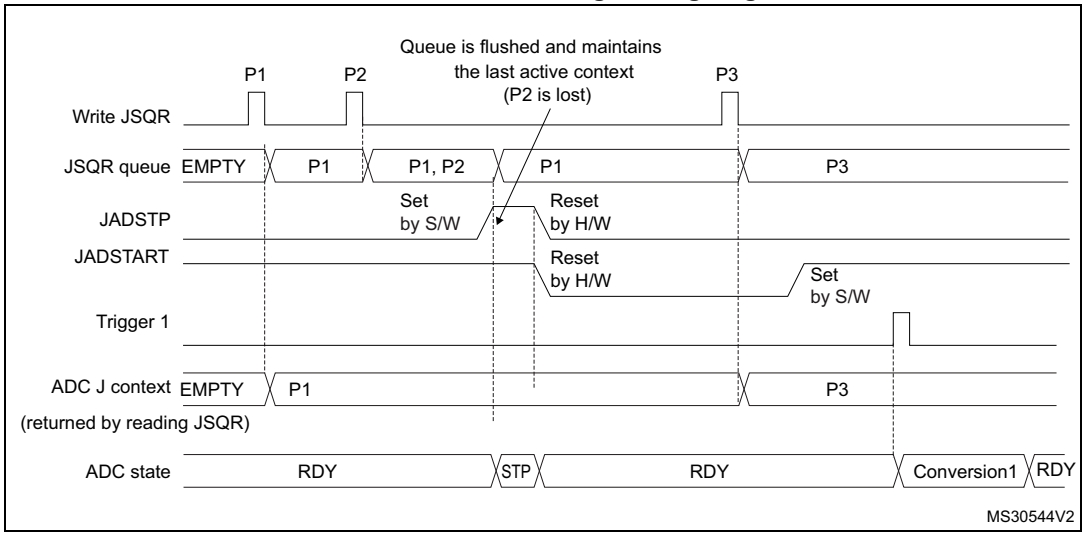


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

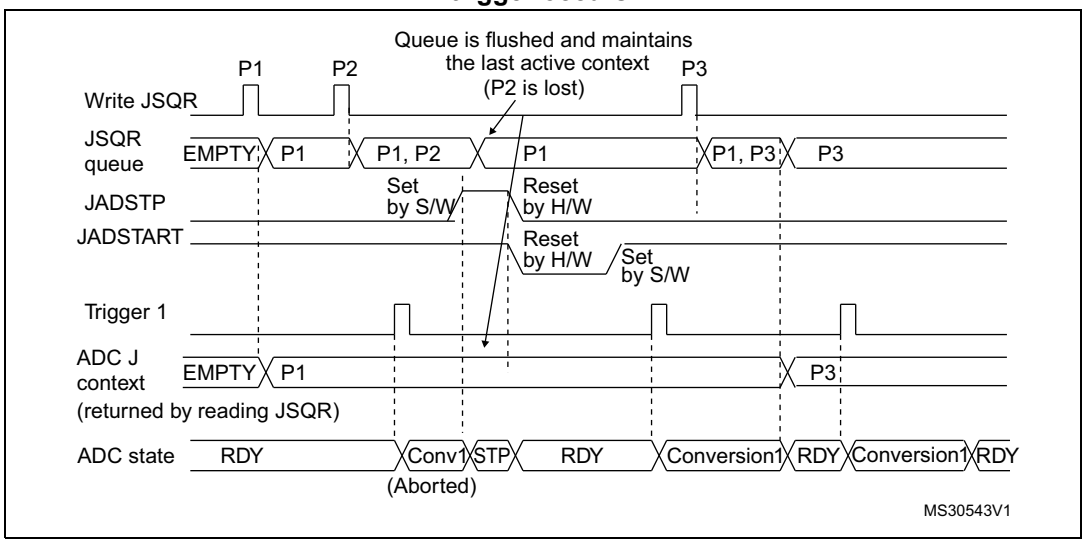
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 110. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion.



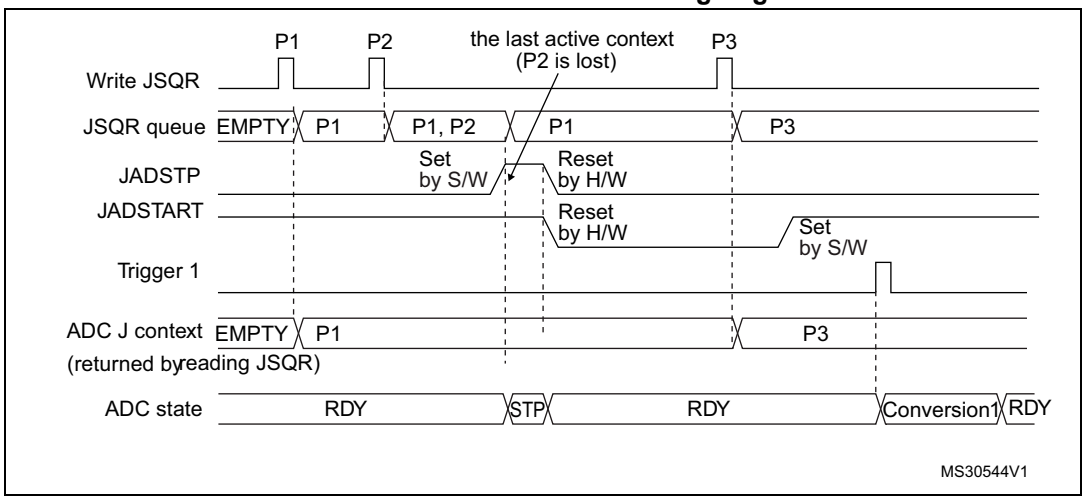
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 111. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.



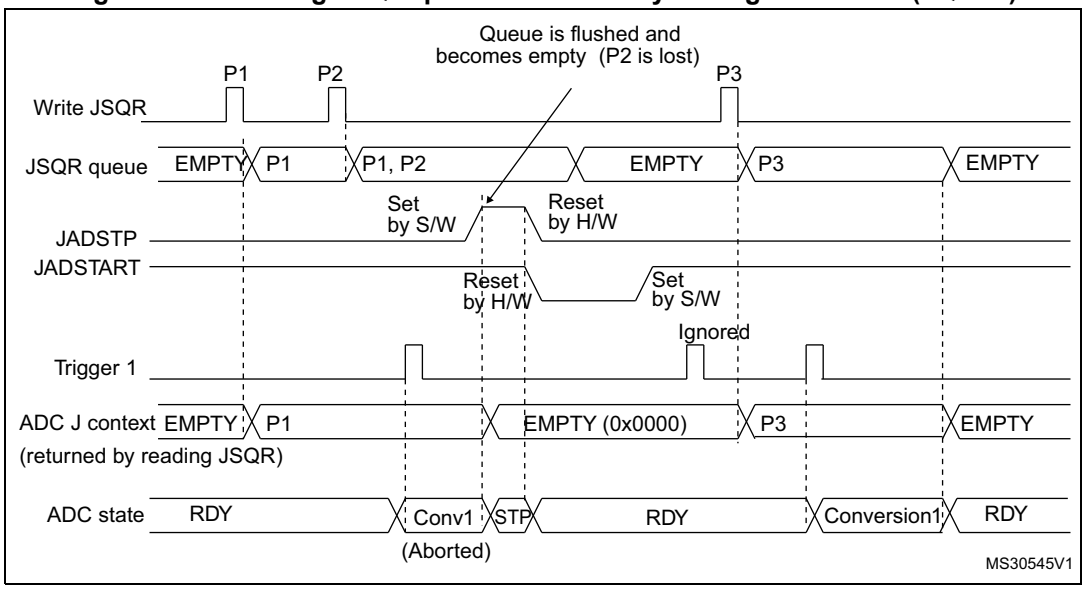
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 112. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion



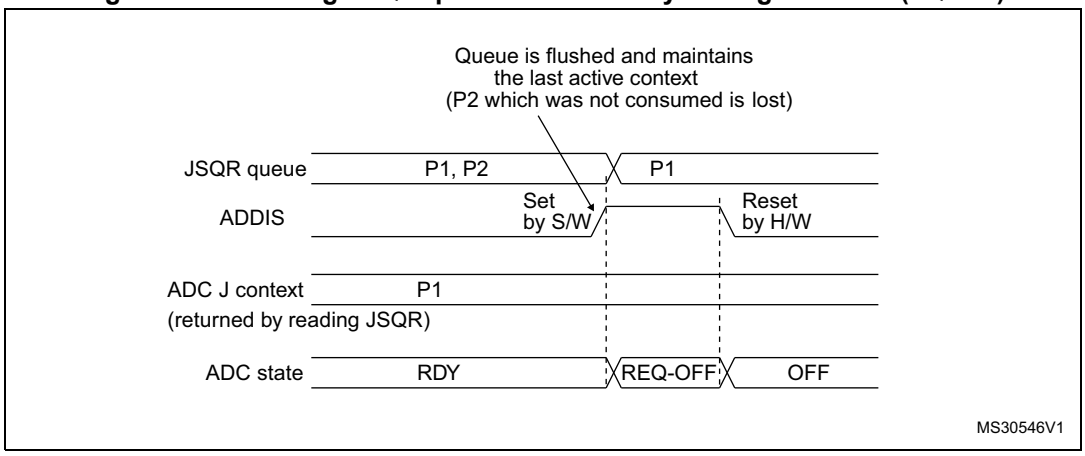
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 113. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1)



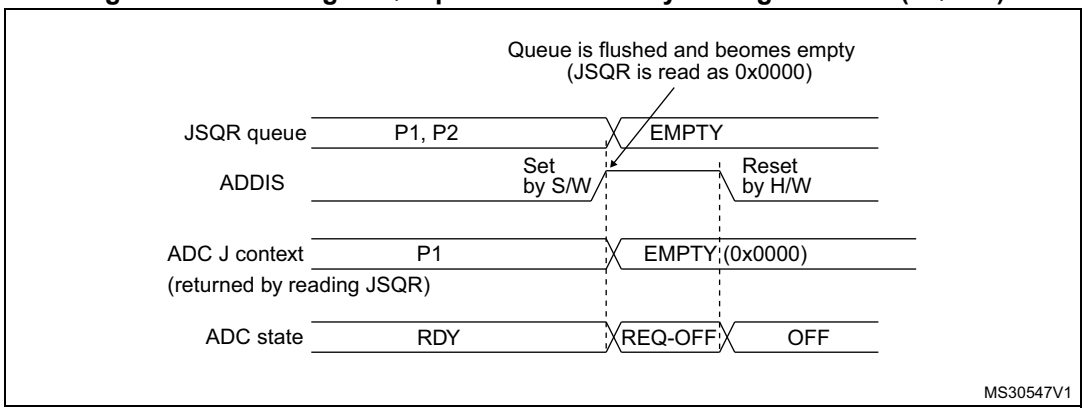
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 114. Flushing JSQR queue of context by setting ADDIS=1 (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

Figure 115. 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 120*, *Figure 121*, *Figure 122* and *Figure 123* 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 163*.

Table 163. 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)

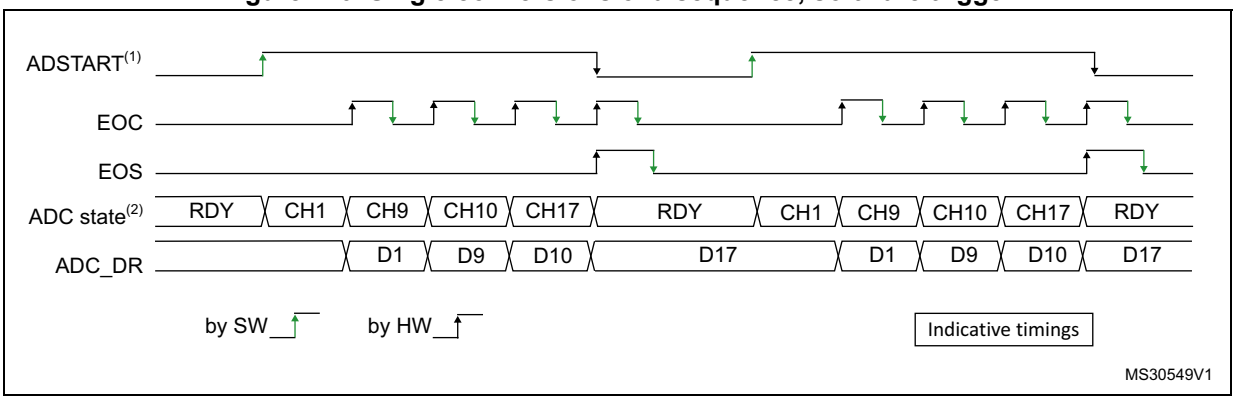
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

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.

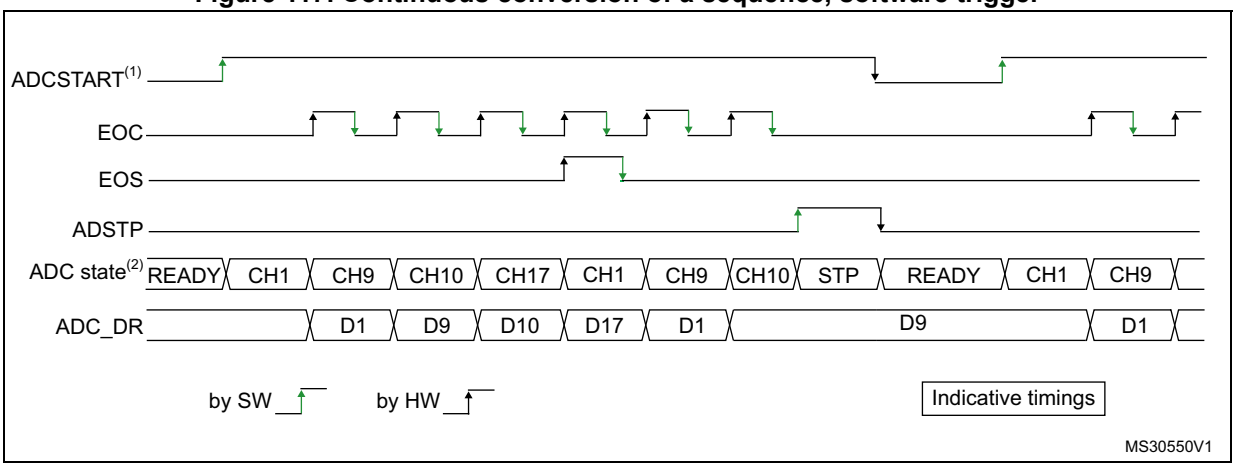
21.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 116. Single conversions of a sequence, software trigger



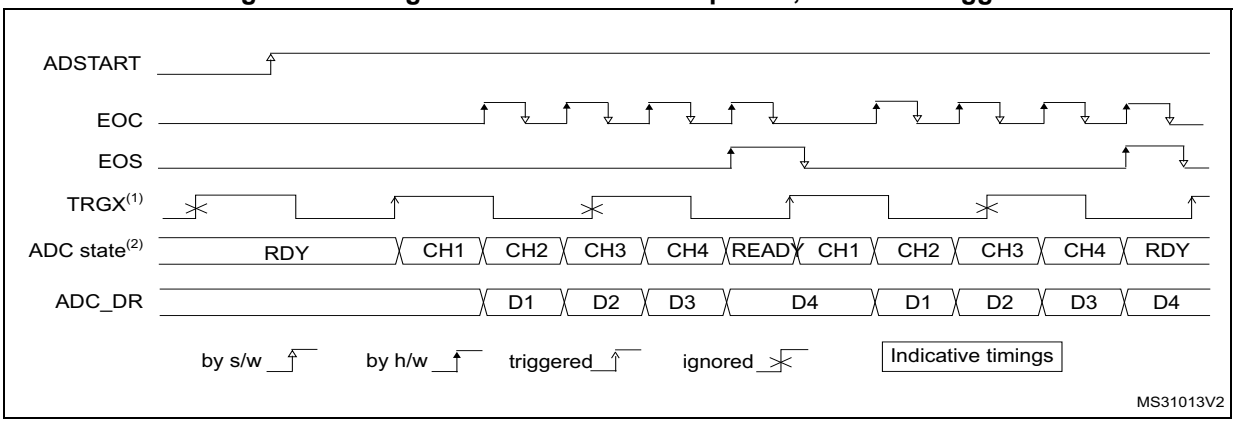
1. EXTEN[1:0]=00, CONT=0
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 117. Continuous conversion of a sequence, software trigger



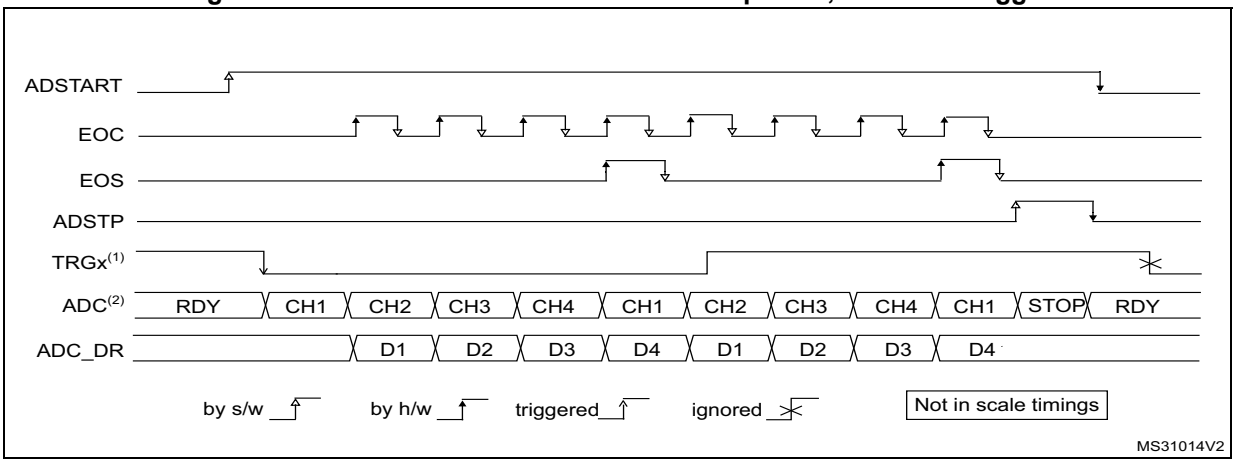
1. EXTEN[1:0]=00, CONT=1
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 118. 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 119. 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 120*, *Figure 121*, *Figure 122* and *Figure 123*.

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 122* and *Figure 123*.

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 166 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 164. 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 164. 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 120, *Figure 121*, *Figure 122* and *Figure 123* show alignments for signed and unsigned data.

Figure 120. Right alignment (offset disabled, unsigned value)

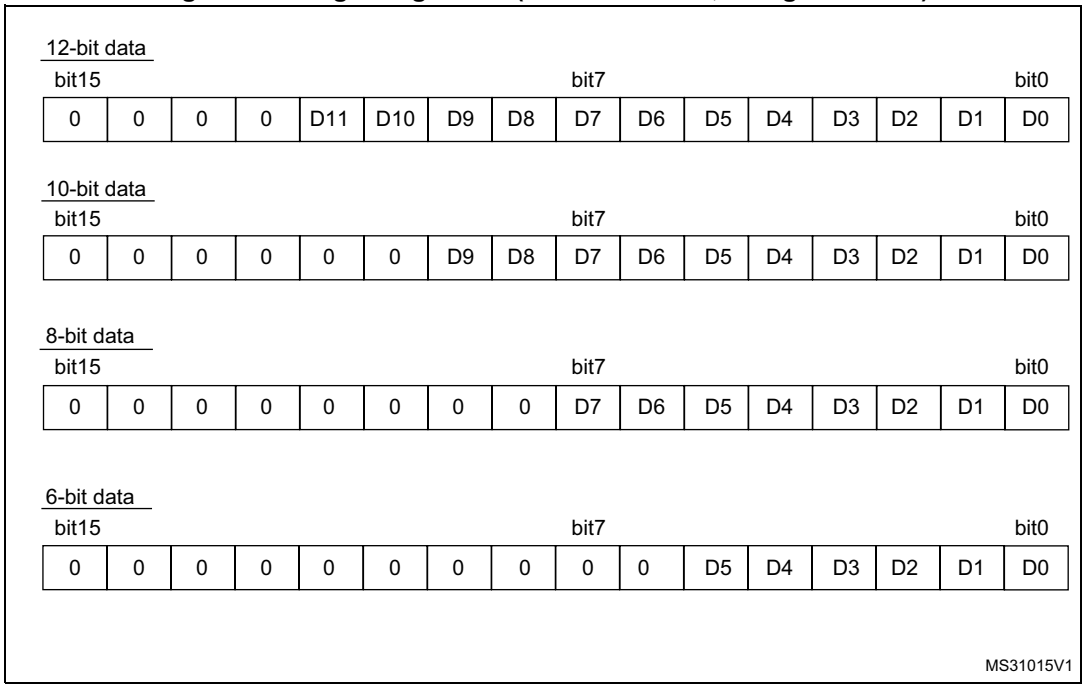


Figure 121. Right alignment (offset enabled, signed value)

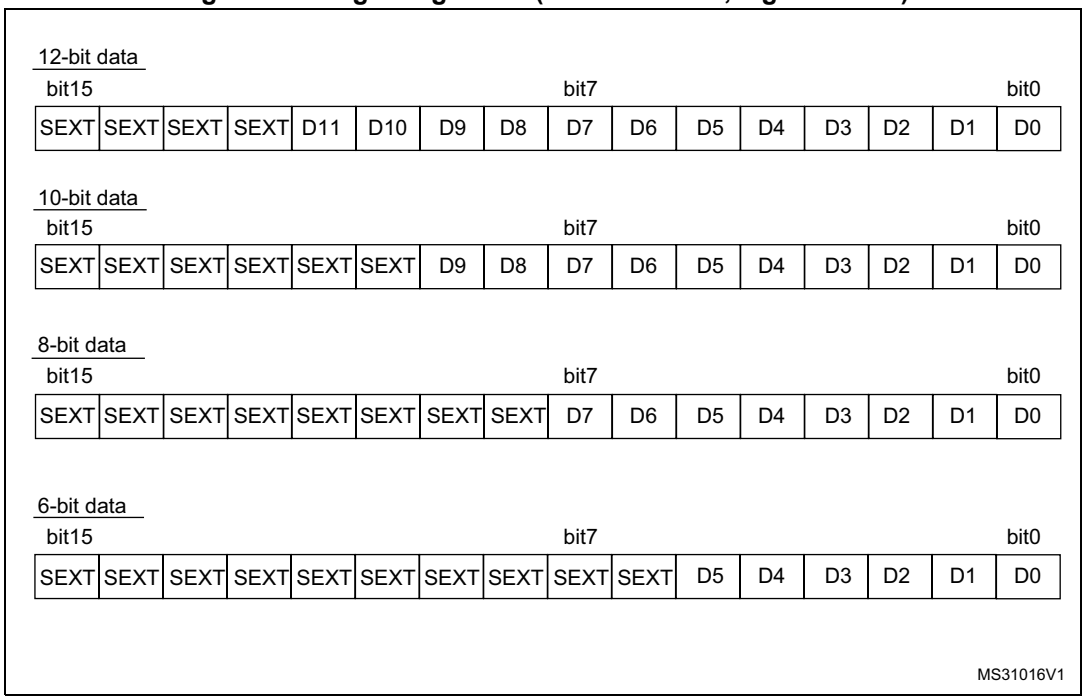


Figure 122. Left alignment (offset disabled, unsigned value)

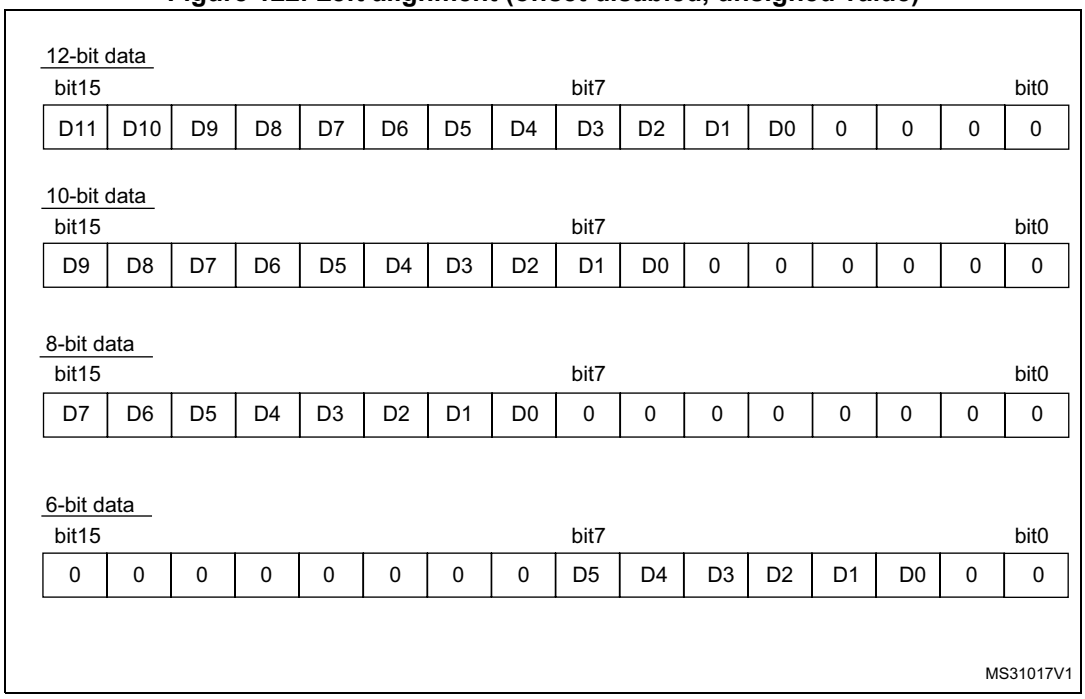
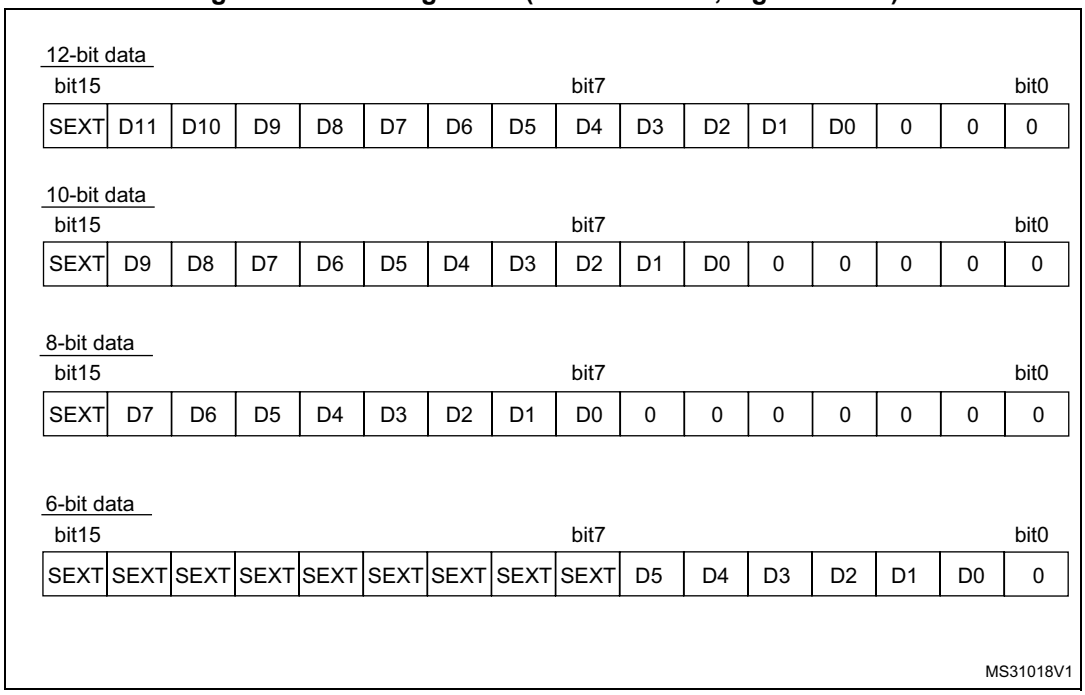


Figure 123. Left alignment (offset enabled, signed value)



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.

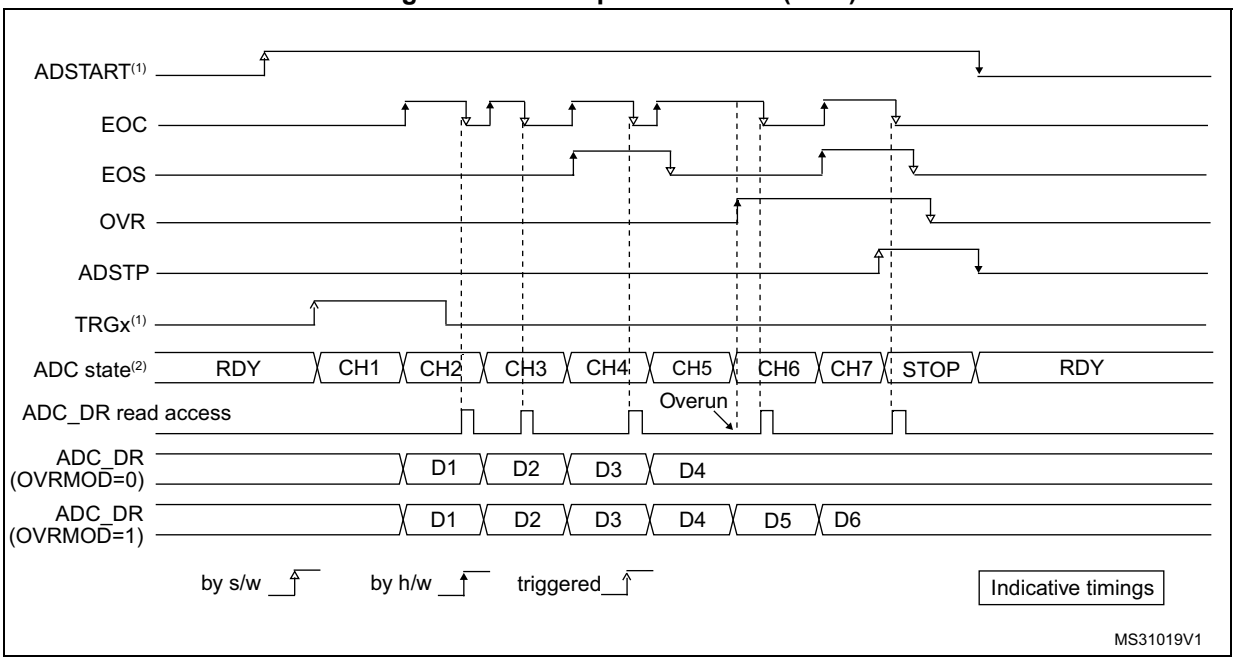
When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting bit ADSTP=1.

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 124. 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 Managing conversions using the DFSDM

The ADC conversion results can be transferred directly to the Digital filter for sigma delta modulators (DFSDM).

In this case, the DFSDMCFG bit must be set to 1 and DMAEN bit must be cleared to 0.

The ADC transfers all the 16 bits of the regular data register to the DFSDM and resets the EOC flag once the transfer is complete.

The data format must be 16-bit signed:

ADC_DR[15:12] = sign extended

ADC_DR[11] = sign

ADC_DR[11:0] = data

To obtain 16-bit signed format in 12-bit ADC mode, the software needs to configure the OFFSETy[11:0] to 0x800 after having set OFFSETy_EN to 1.

Only right aligned data format is available for the DFSDM interface (see *Figure 121: Right alignment (offset enabled, signed value)*).

21.4.28 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 125*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 126*).

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 126*).
- 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 128*).

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 129*).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO=1, CONT=1 and AUTDLY=1), follow the following procedure:

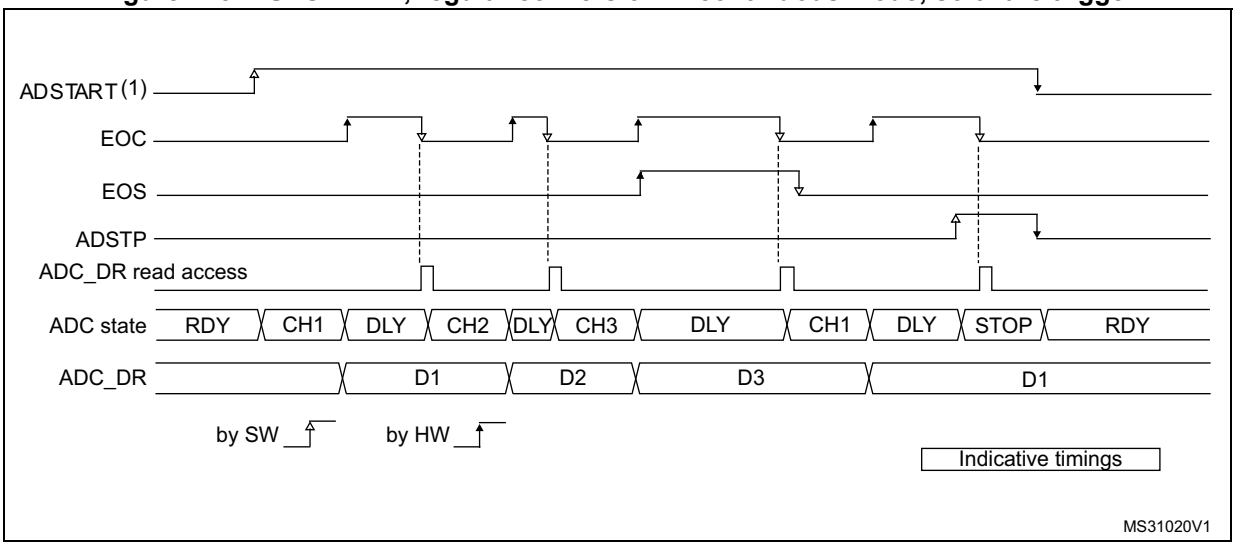
1. Wait until JEOS=1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP=1
4. Read the regular data.

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.

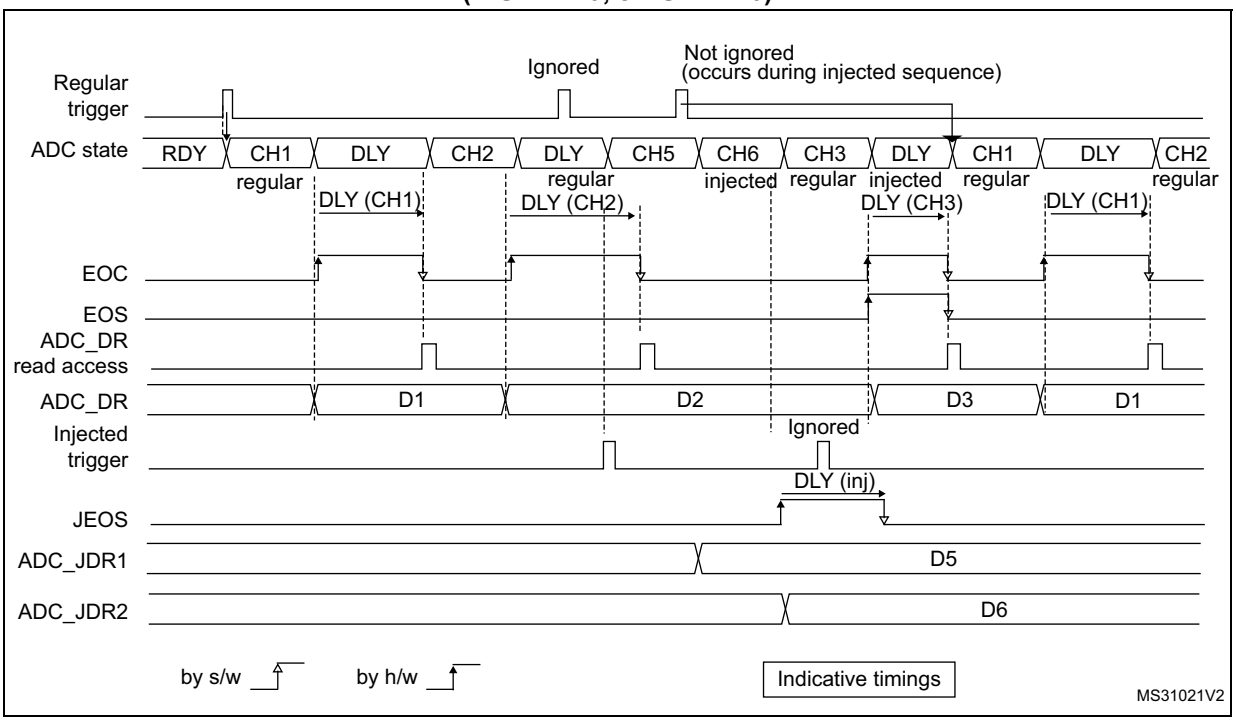
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 125. AUTODLY=1, regular conversion in continuous mode, software trigger



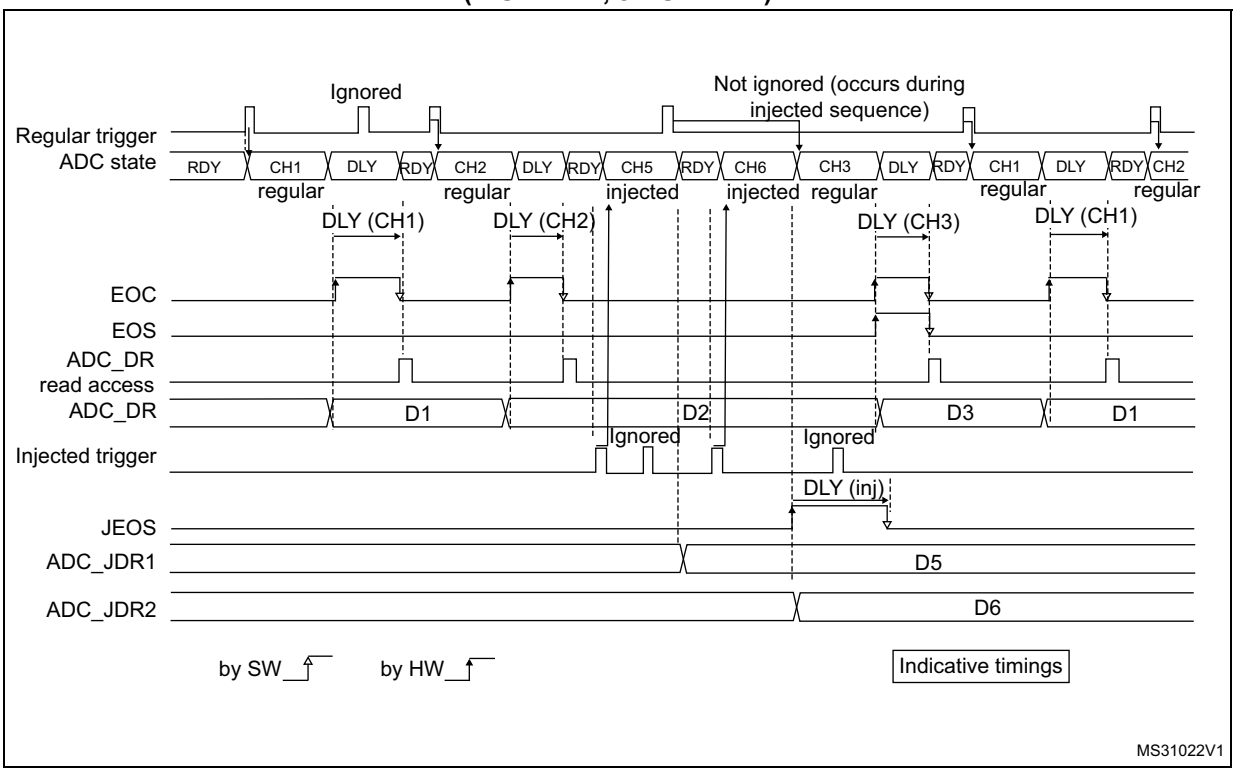
1. AUTDLY=1
2. Regular configuration: EXTEN[1:0]=00 (SW trigger), CONT=1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

Figure 126. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)



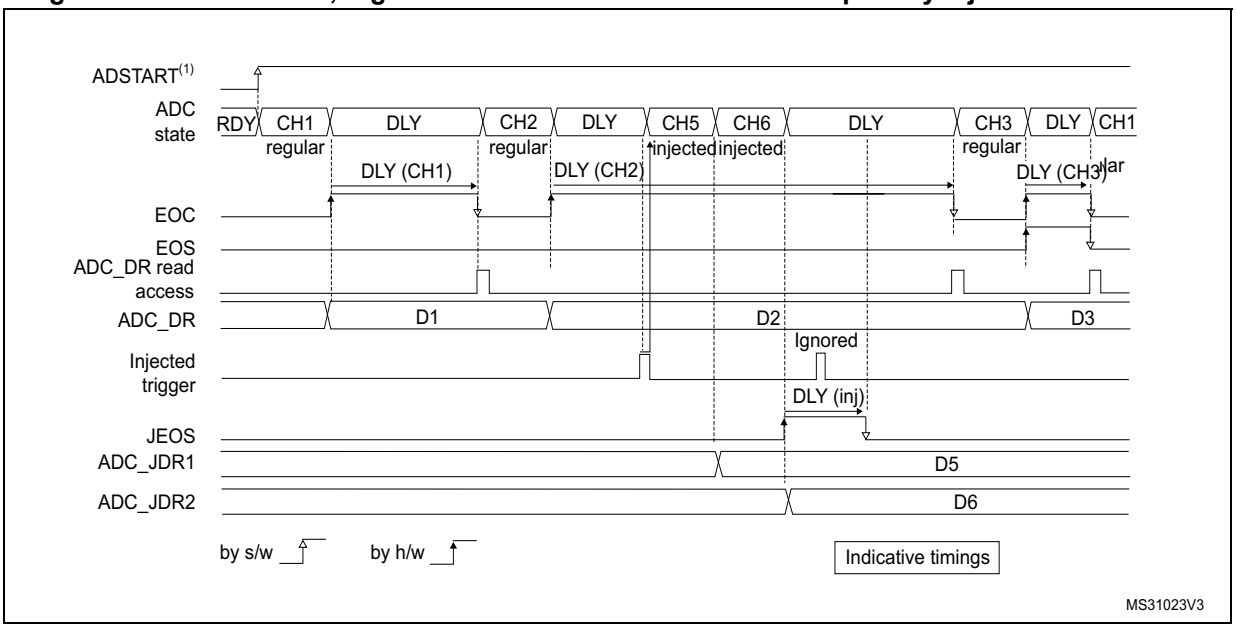
1. AUTDLY=1
2. Regular configuration: EXTEN[1:0]=01 (HW trigger), CONT=0, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN[1:0]=01 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 127. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)



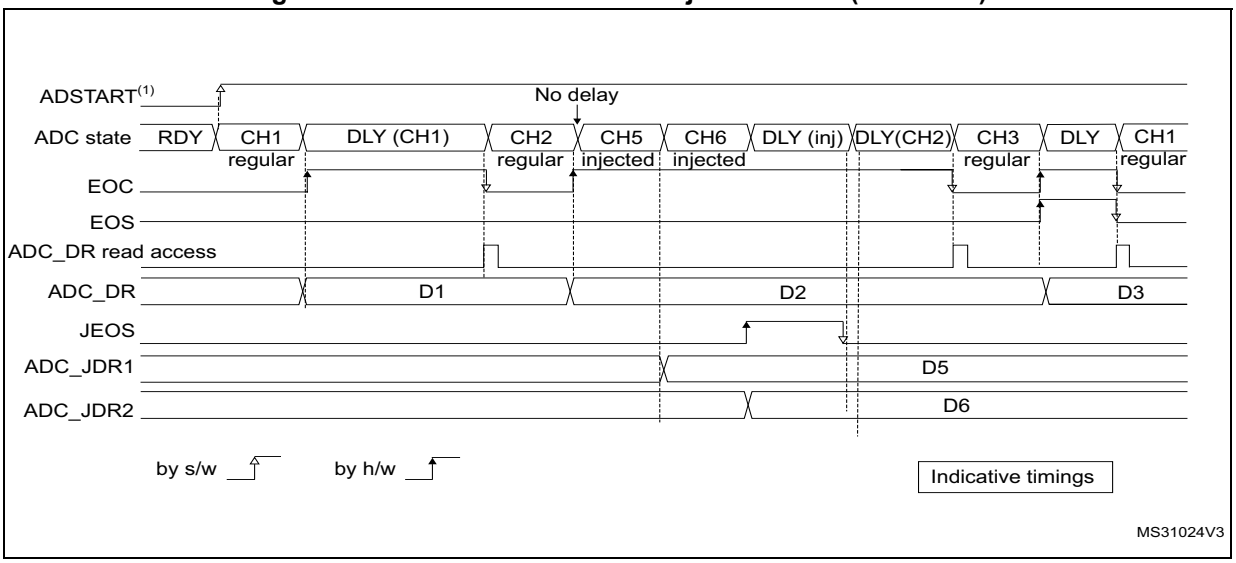
1. AUTDLY=1
2. Regular configuration: EXTEN[1:0]=01 (HW trigger), CONT=0, DISCEN=1, DISCNUM=1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0]=01 (HW Trigger), JDISCEN=1, CHANNELS = 5,6

Figure 128. 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 129. AUTODLY=1 in auto- injected mode (JAUTO=1)

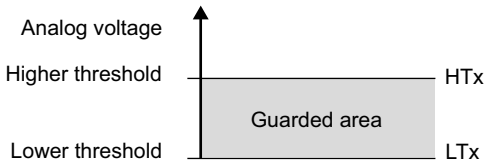


1. AUTDLY=1
2. Regular configuration: EXTEN[1:0]=00 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2
3. Injected configuration: JAUTO=1, CHANNELS = 5,6

21.4.29 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 130. 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 165 shows how the ADC_CFGR registers should be configured to enable the analog watchdog on one or more channels.

Table 165. 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 166 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 166. Analog watchdog 1 comparison

Resolution(bit RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	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

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 167* describes how the comparison is performed for all the possible resolutions.

Table 167. Analog watchdog 2 and 3 comparison

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	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

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 131. ADCy_AWDx_OUT signal generation (on all regular channels)

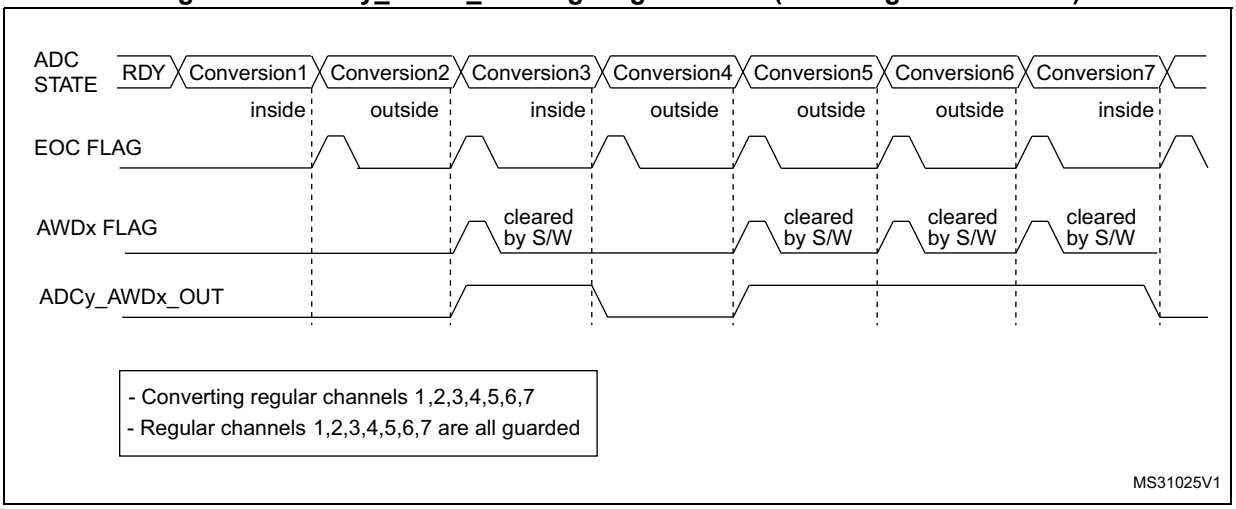


Figure 132. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software)

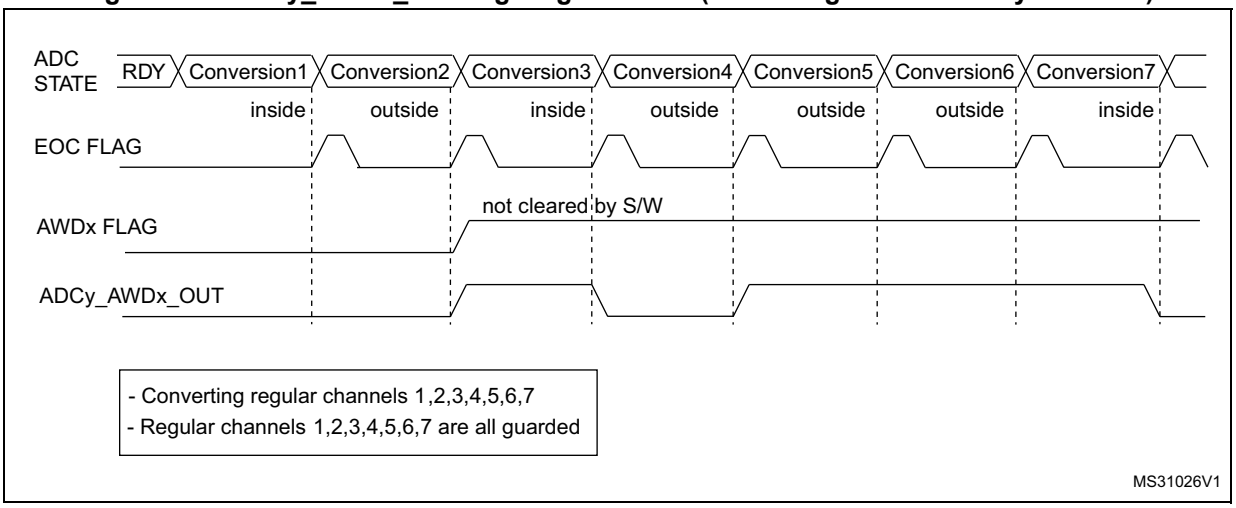


Figure 133. ADCy_AWDx_OUT signal generation (on a single regular channel)

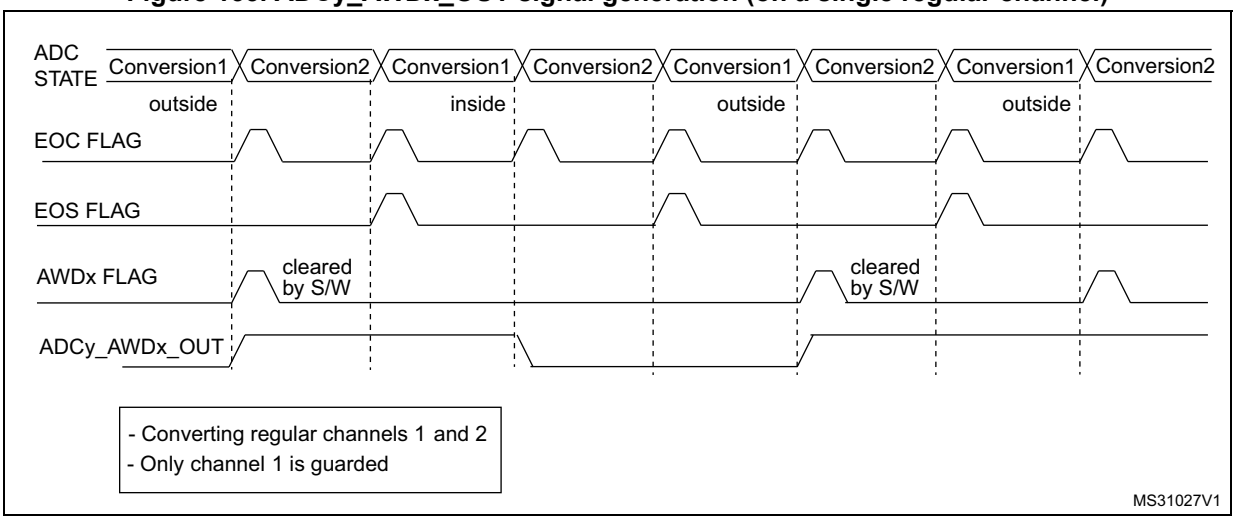
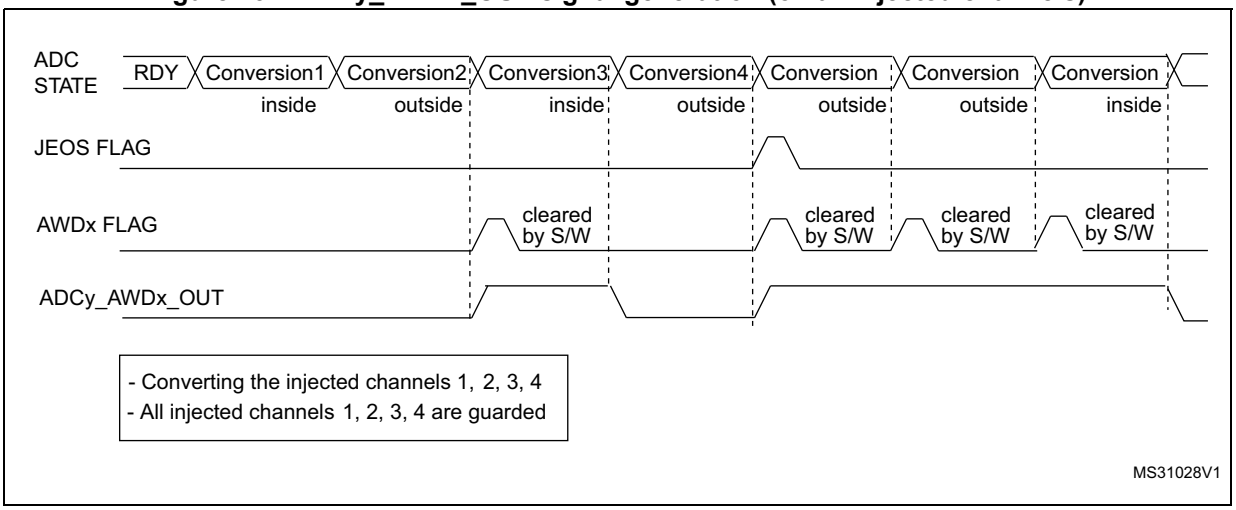


Figure 134. ADCy_AWDx_OUT signal generation (on all injected channels)



21.4.30 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 135. 20-bit to 16-bit result truncation

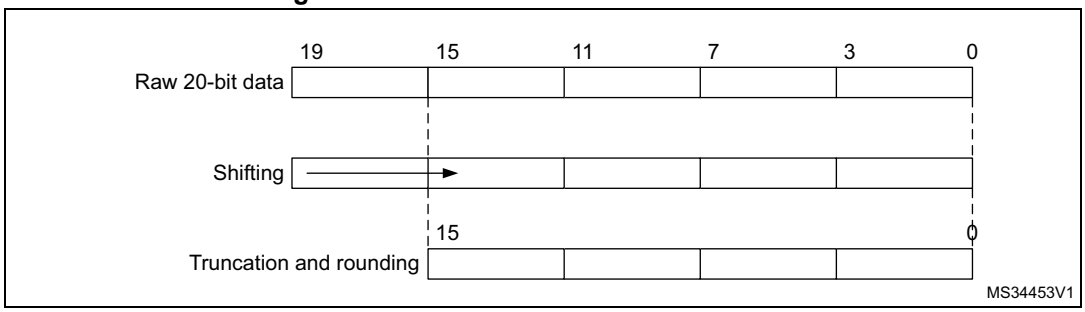


Figure 136 gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 136. Numerical example with 5-bit shift and rounding

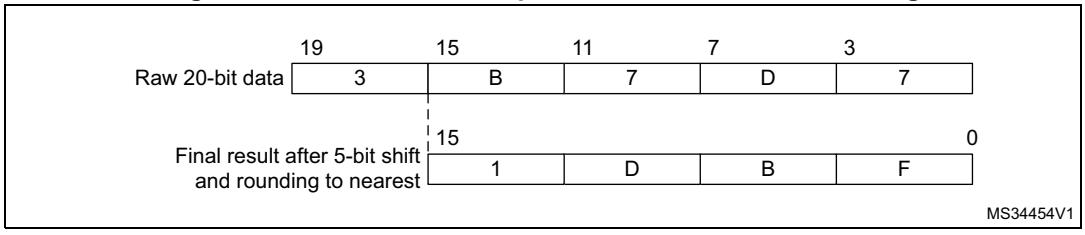


Table 168 gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 168. 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.

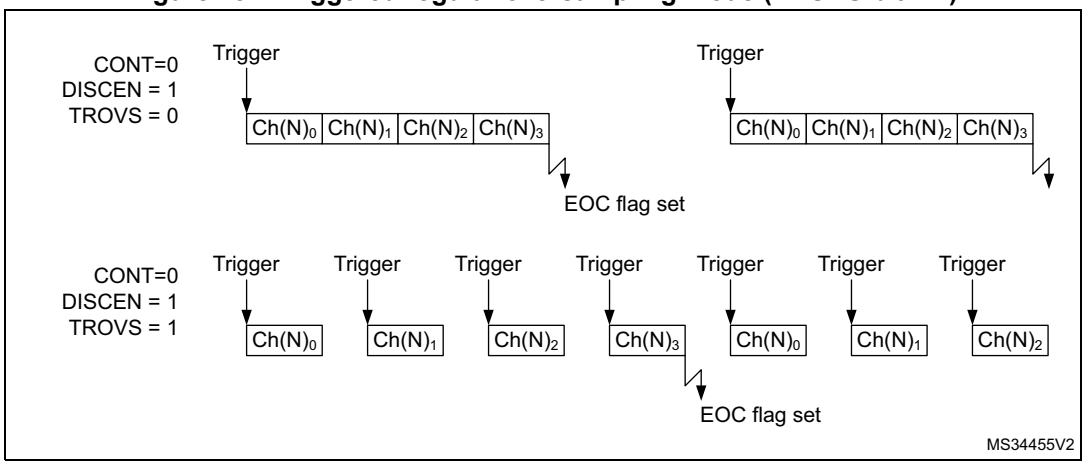
Triggered mode

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 137 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 137. 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).

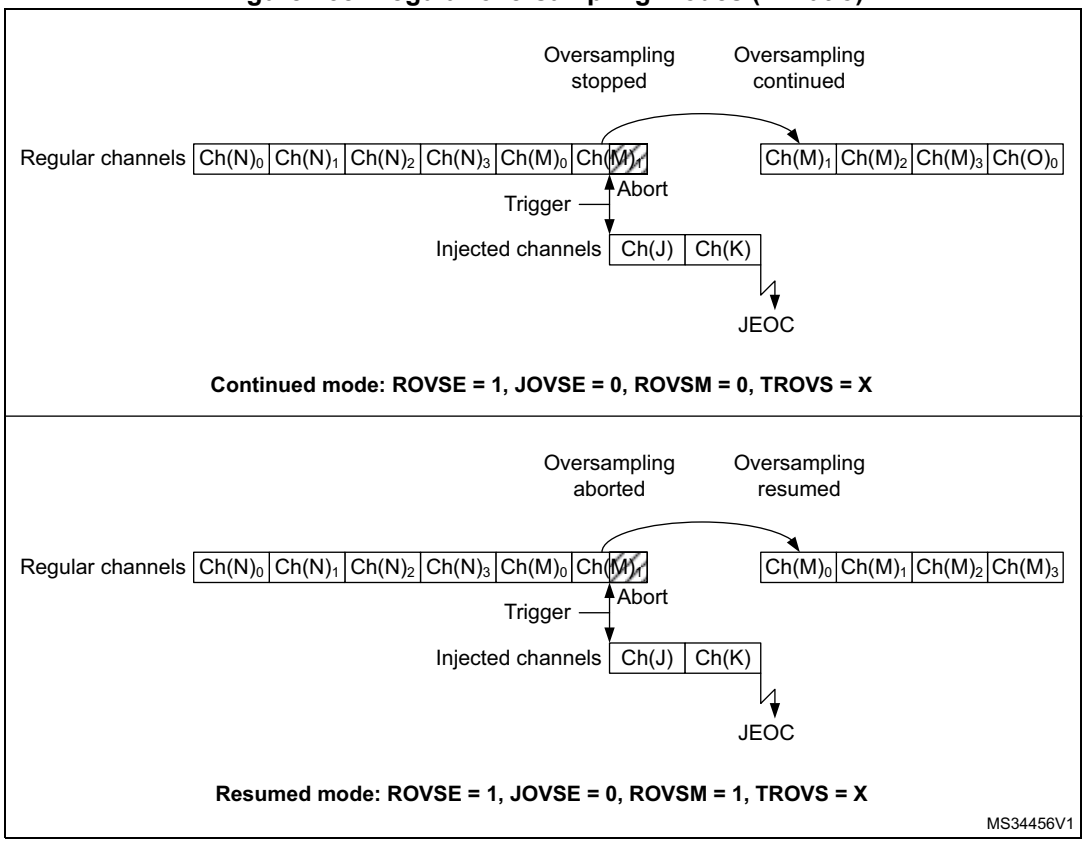
Oversampling regular channels only

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 138 gives examples for a 4x oversampling ratio.

Figure 138. Regular oversampling modes (4x ratio)



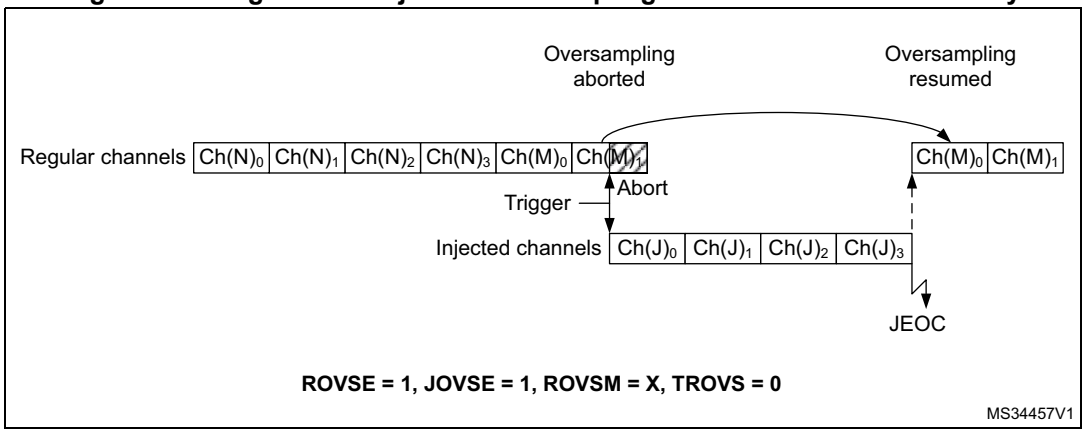
Oversampling Injected channels only

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 139* below.

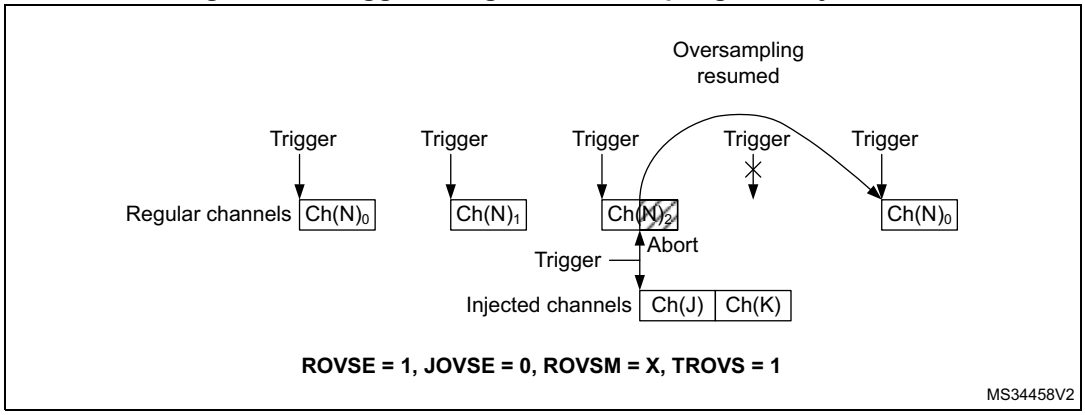
Figure 139. Regular and injected oversampling modes used simultaneously



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 140* below.

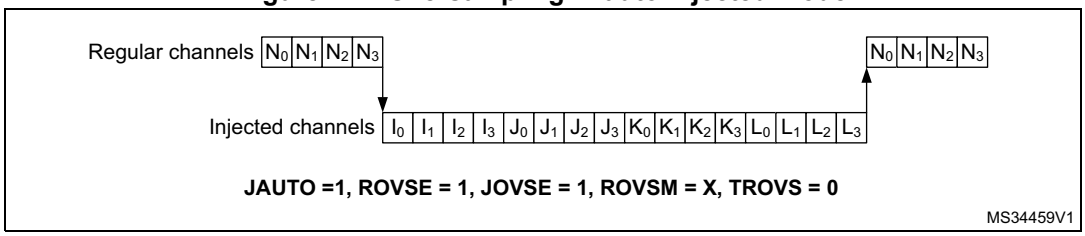
Figure 140. Triggered regular oversampling with injection



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 141* below shows how the conversions are sequenced.

Figure 141. Oversampling in auto-injected mode



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 169* below summarizes all combinations, including modes not supported.

Table 169. 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

21.4.31 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see *Figure 142*).

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.

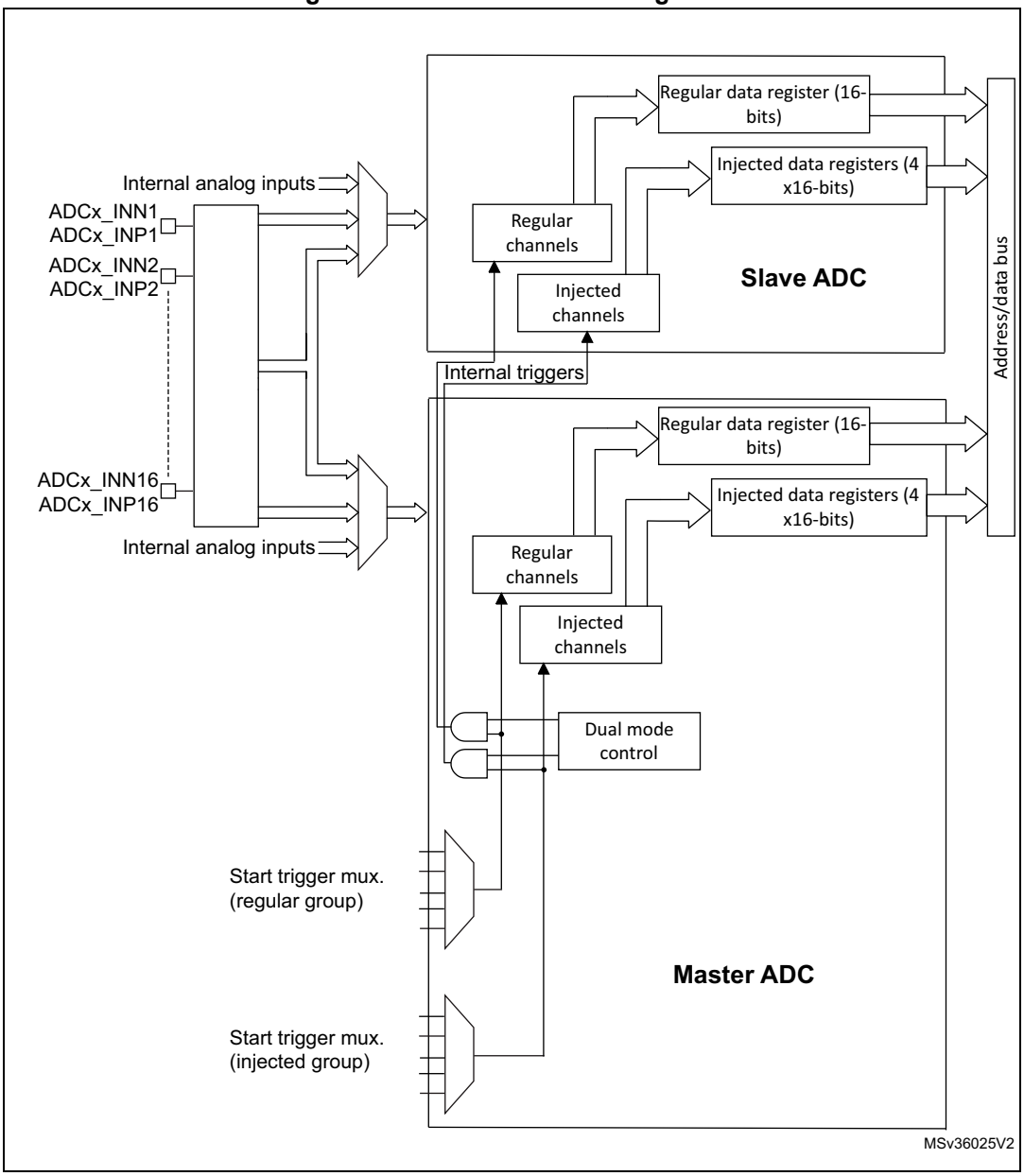
To start a conversion in dual mode, the user must program the bits EXTEN[1:0], EXTSEL, JEXTEN[1:0], JEXTSEL of the master ADC only, to configure a software or hardware trigger, and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don't care).

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 142. 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).

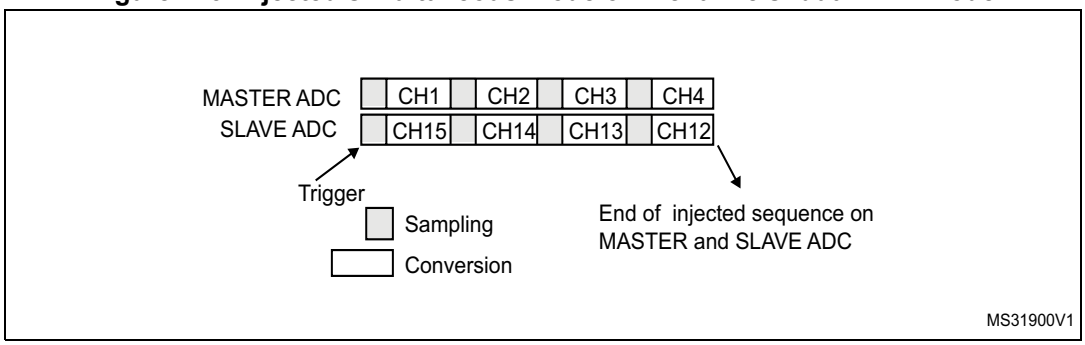
Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

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 143*), 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 143. 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:

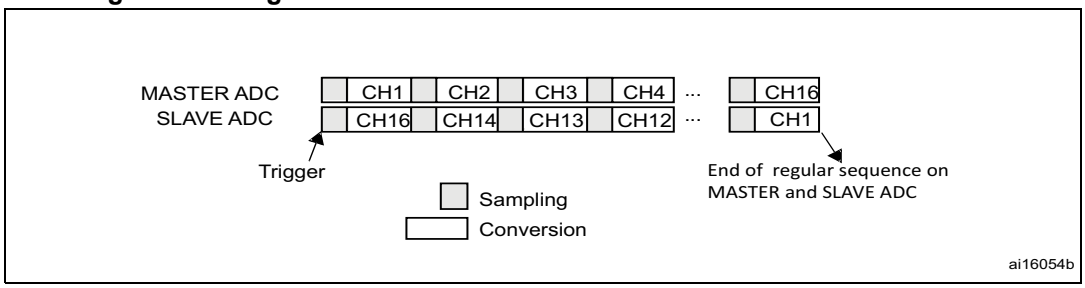
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software 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 144*), 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 144. 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 145. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode

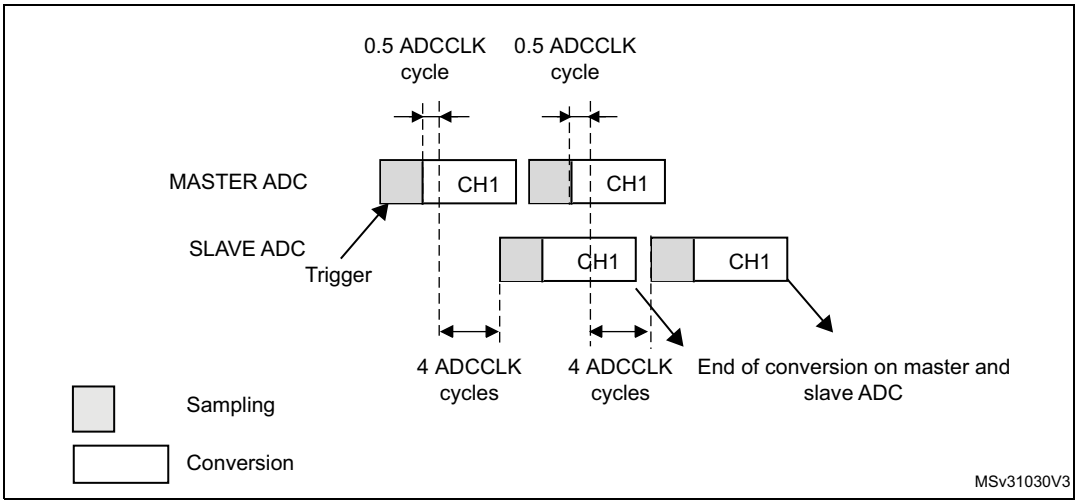
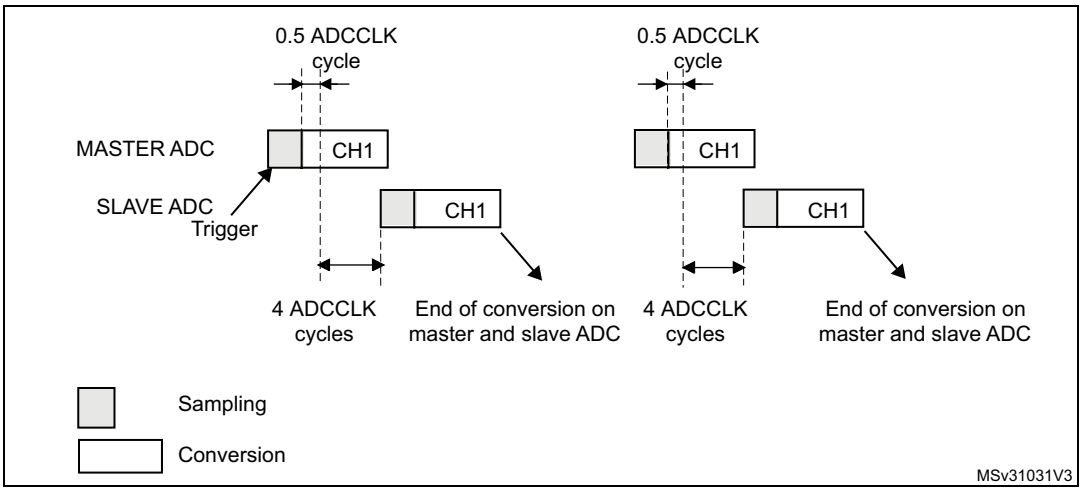


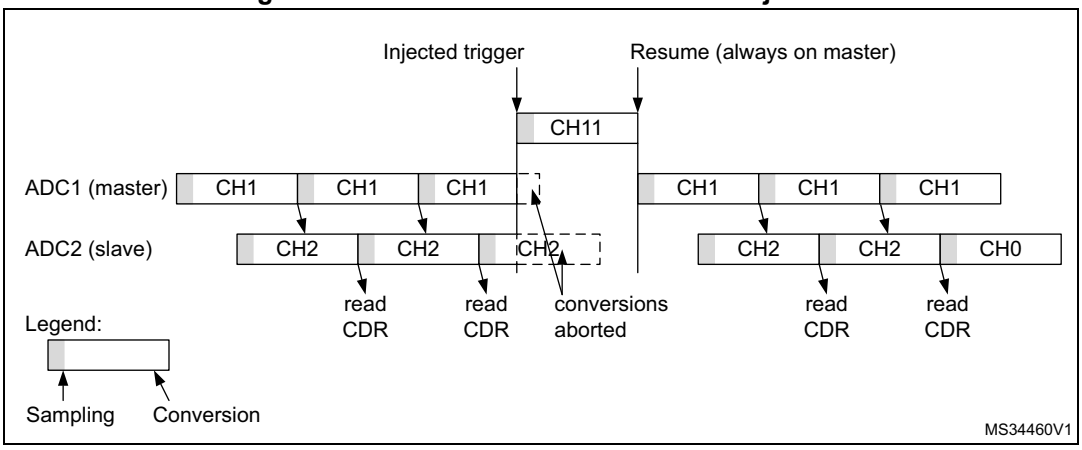
Figure 146. Interleaved mode on 1 channel in single conversion mode: dual ADC mode



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 147* below).

Figure 147. 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.

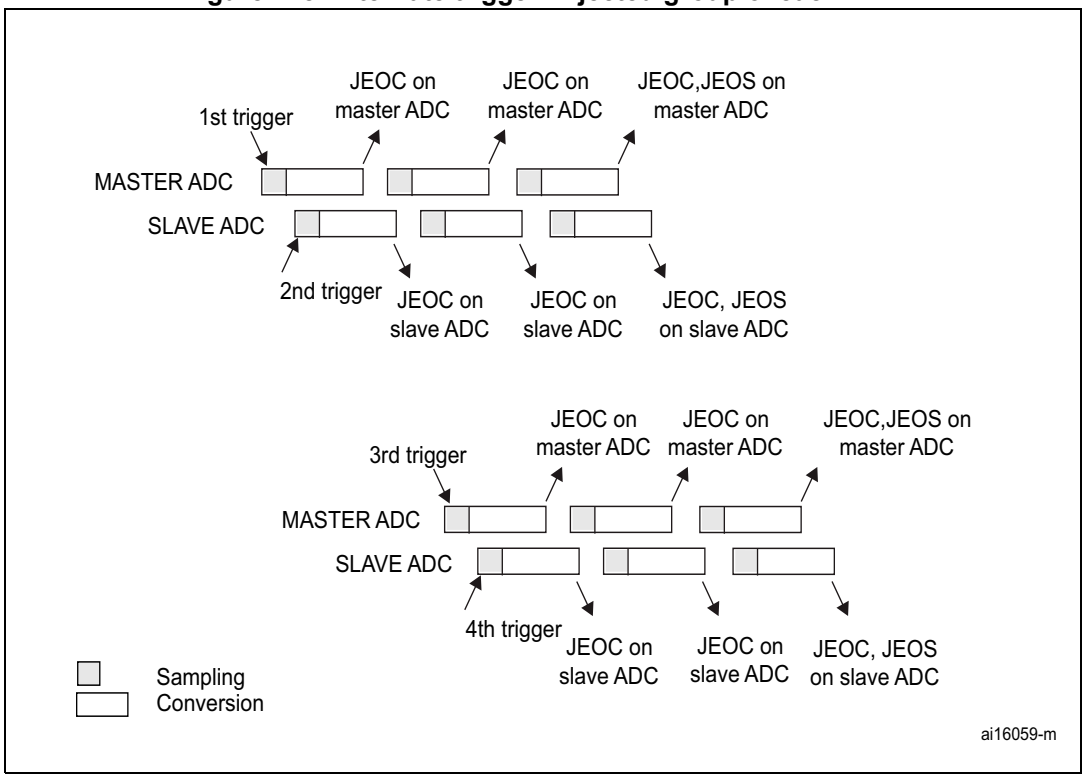
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

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 148. 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.

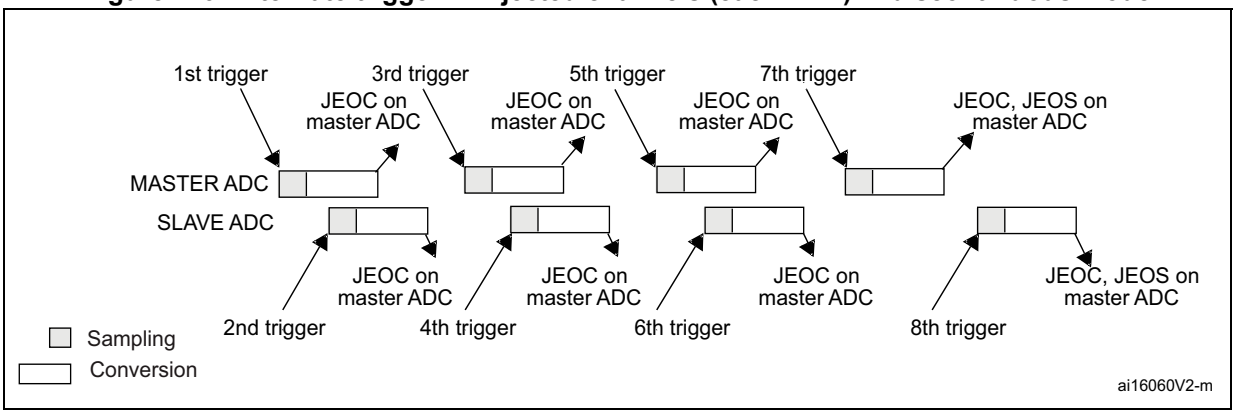
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

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 149. 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

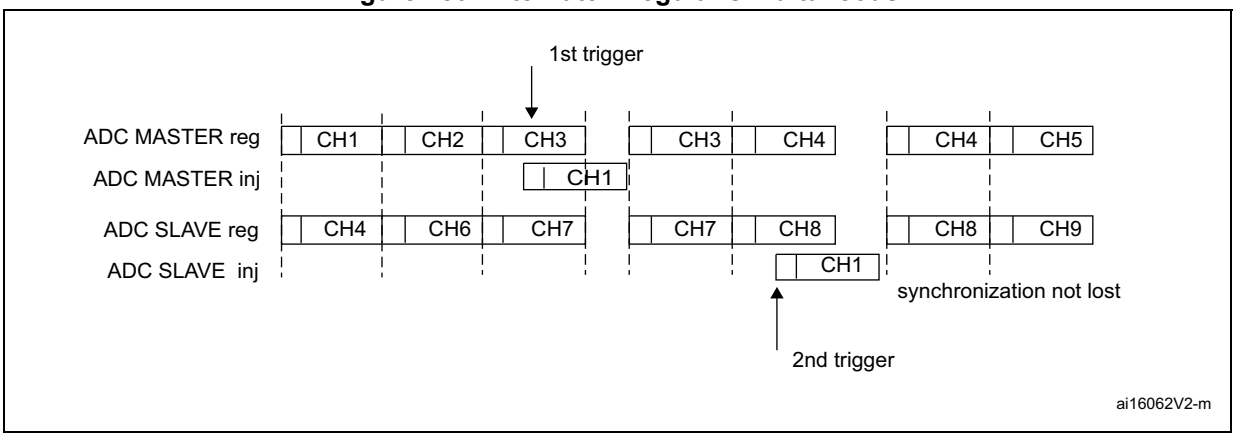
This mode is selected by programming bits DUAL[4:0]=00010.

It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. *Figure 150* 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.

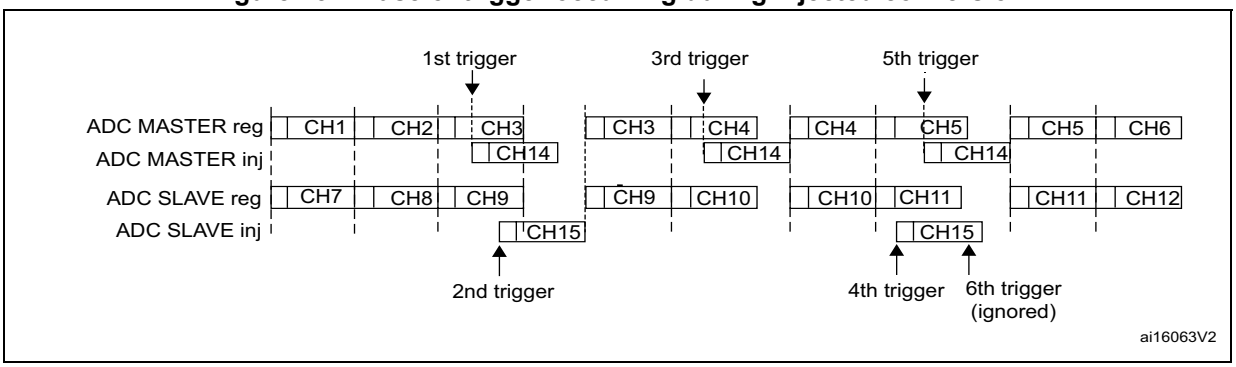
Note: In combined regular simultaneous + alternate trigger mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Figure 150. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 151* shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

Figure 151. 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 152*, *Figure 153* and *Figure 154* 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 152. Interleaved single channel CH0 with injected sequence CH11, CH12

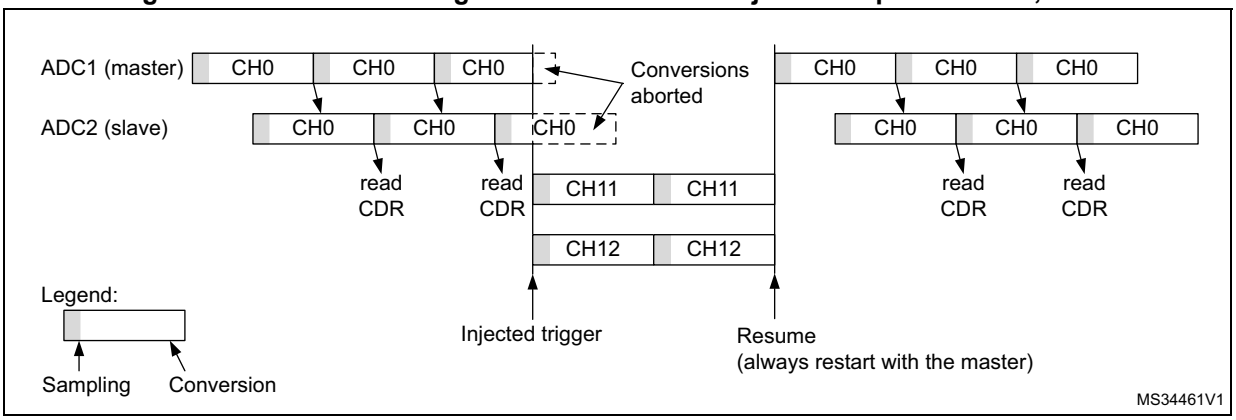


Figure 153. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first

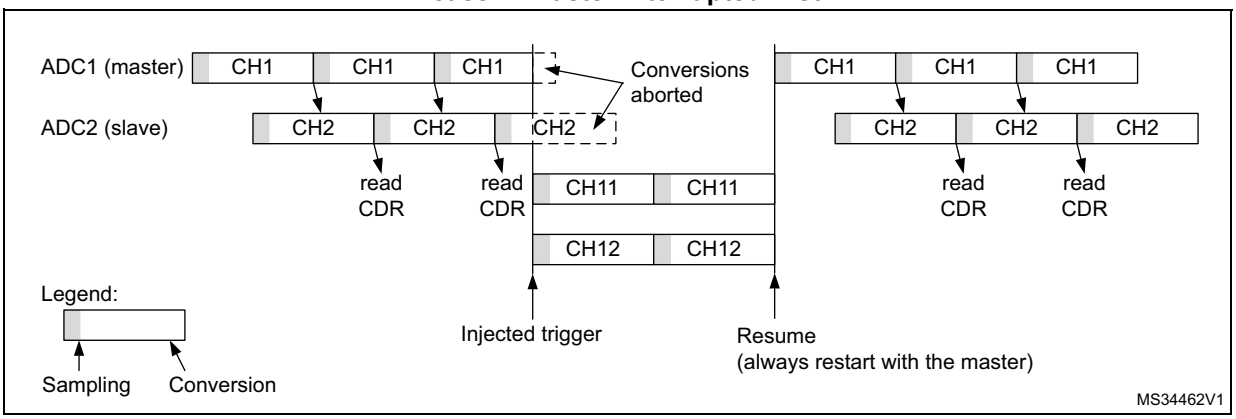
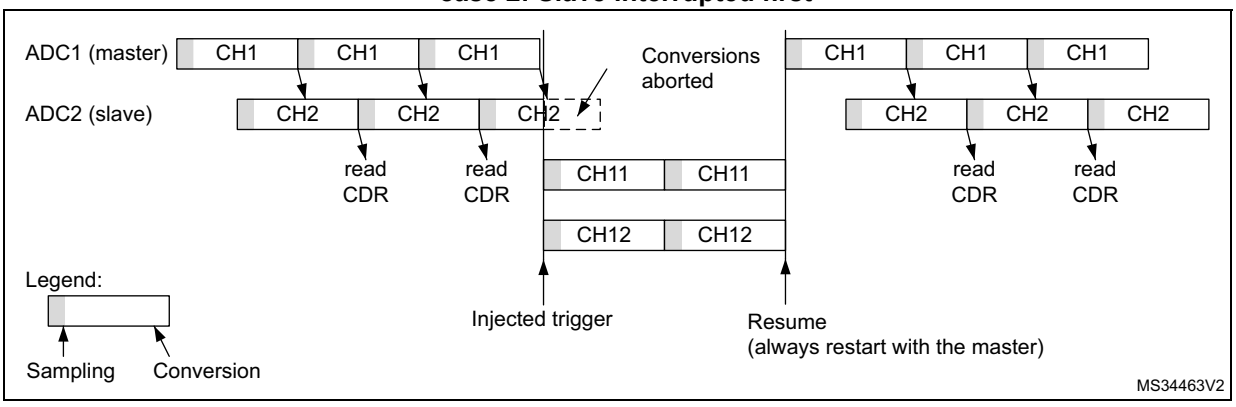


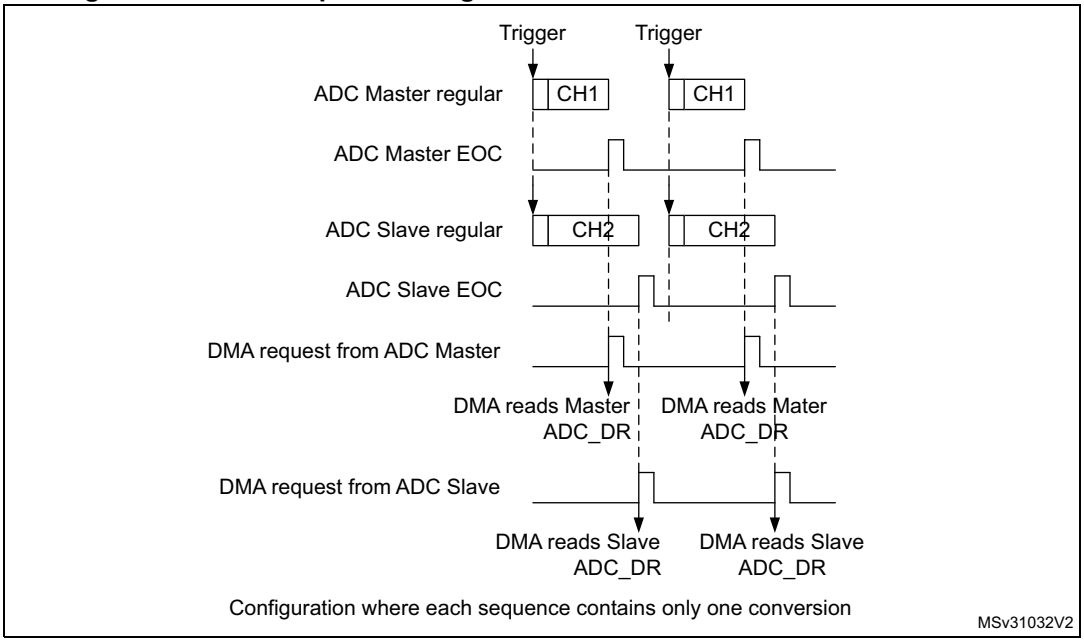
Figure 154. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first



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 155: DMA Requests in regular simultaneous mode when MDMA=0b00*).

Figure 155. 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 156. DMA requests in regular simultaneous mode when MDMA=0b10

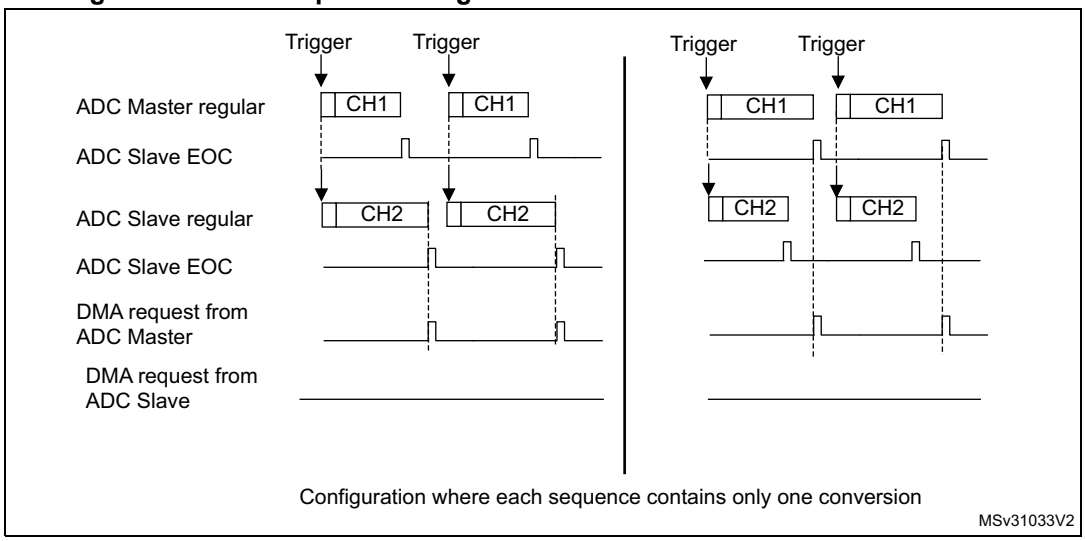
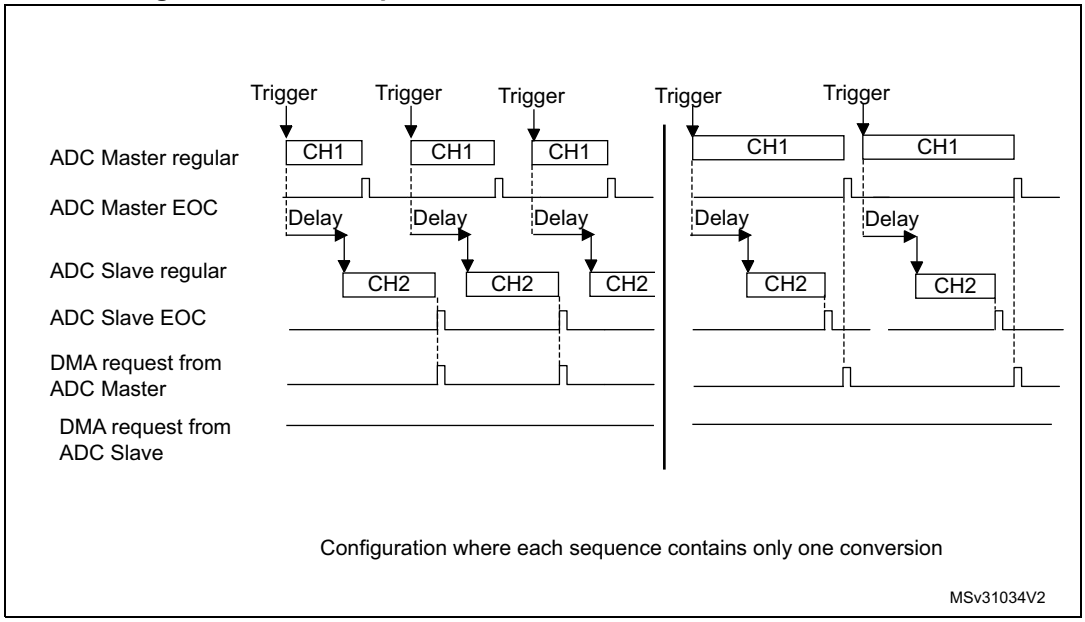


Figure 157. DMA requests in interleaved mode when MDMA=0b10



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.

21.4.32 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 158 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.

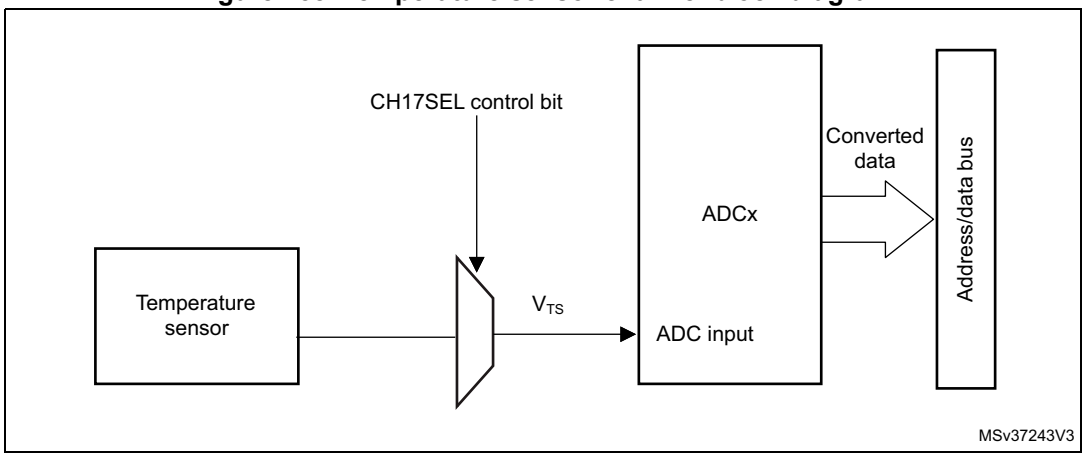
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet 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 158 shows the block diagram of the temperature sensor.

Figure 158. 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 CH17SEL 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 CH17SEL 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).

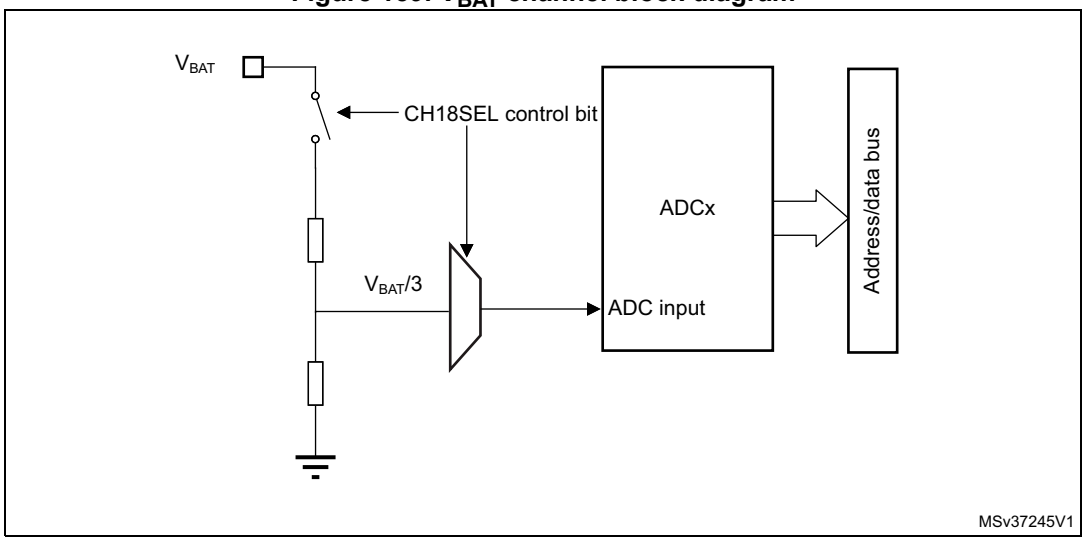
21.4.33 V_{BAT} supply monitoring

The CH18SEL 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 CH18SEL 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 159. V_{BAT} channel block diagram



1. The CH18SEL bit must be set to enable the conversion of internal channel for V_{BAT}/3.

21.4.34 Monitoring the internal voltage reference

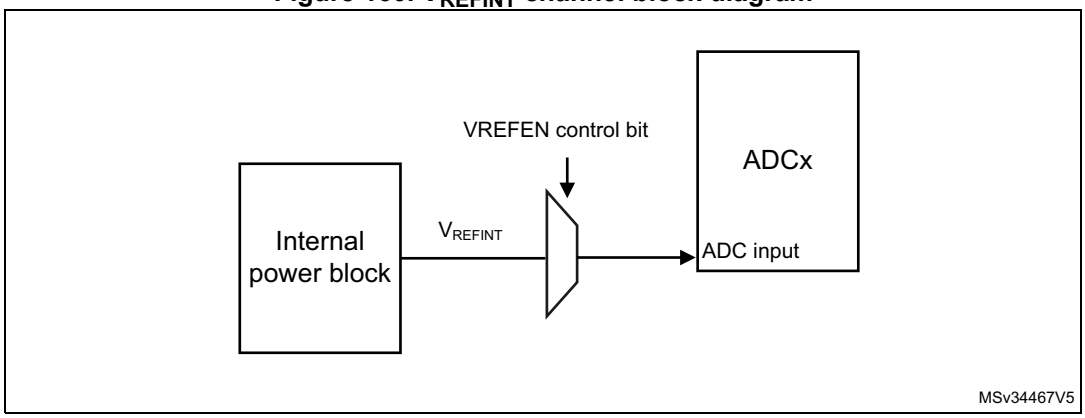
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

The internal voltage reference is internally connected to the input channel 0 of the ADC1 (ADC1_INP0).

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 160 shows the block diagram of the V_{REFINT} sensing feature.

Figure 160. 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 170. 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 76* 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.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
					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).

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	JAWD1 EN	AWD1 EN	AWD1S GL	JQM	JDISC EN	DISCNUM[2:0]			DISC EN
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.	AUT DLY	CONT	OVR MOD	EXTEN[1:0]		EXTSE L3	EXTSE L2	EXTSE L1	EXTSE L0	ALIGN	RES[1:0]		DFSD MCFG	DMA CFG	DMA EN
	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 Reserved, must be kept at reset value.

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 9:6 **EXTSEL[3:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

0000: Event 0

0001: Event 1

0010: Event 2

0011: Event 3

0100: Event 4

0101: Event 5

0110: Event 6

0111: Event 7

...

1111: Event 15

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 **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).

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 **DFSDMCFG**: DFSDM mode configuration

This bit is set and cleared by software to enable the DFSDM mode. It is effective only when DMAEN=0.

0: DFSDM mode disabled

1: DFSDM mode enabled

Note: To make sure no conversion is ongoing, the software is allowed to write this bit only when ADSTART= 0 and JADSTART= 0.

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	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.

Bits 27:17 Reserved, must be kept at reset value.

Bits 16:11 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
Res.	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
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

Bits 31:30 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.	Res.	Res.	Res.	LT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to *Section 21.4.29: 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 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to *Section 21.4.29: 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 21.4.29: 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 21.4.29: 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 21.4.29: 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
Res.	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:2]		
	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
JSQ2[1:0]		Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[3:0]				JL[1:0]	
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 **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.

Note: The software is allowed to write these bits only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **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 19 Reserved, must be kept at reset value.

Bits 18:14 **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 13 Reserved, must be kept at reset value.

Bits 12:8 **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 7:6 **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 5:2 **JEXTSEL[3:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

0000: Event 0

0001: Event 1

0010: Event 2

0011: Event 3

0100: Event 4

0101: Event 5

0110: Event 6

0111: Event 7

...

1111: Event 15

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

Bits 25:12 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).

21.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

21.7.1 ADC common status register (ADC_CSR)

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.

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 ADC common control register (ADC_CCR)

Address offset: 0x08 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH18SEL	CH17SEL	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 **CH18SEL**: CH18 selection
This bit is set and cleared by software to control channel 18.
0: V_{BAT} channel disabled.
1: V_{BAT} channel enabled

Bit 23 **CH17SEL**: CH17 selection
This bit is set and cleared by software to control channel 17.
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: CK_ADCx (x=123) (Asynchronous clock mode), generated at product level (refer to *Section 6: Reset and clock control (RCC)*)

01: 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: HCLK/2 (Synchronous clock mode)

11: 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: Enable dual interleaved mode to output to the master channel of DFSDM interface both Master and the Slave result (16-bit data width)
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 171* 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 171. DELAY bits versus ADC resolution

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * T_{ADC_CLK}	1 * T_{ADC_CLK}	1 * T_{ADC_CLK}	1 * T_{ADC_CLK}
0001	2 * T_{ADC_CLK}	2 * T_{ADC_CLK}	2 * T_{ADC_CLK}	2 * T_{ADC_CLK}
0010	3 * T_{ADC_CLK}	3 * T_{ADC_CLK}	3 * T_{ADC_CLK}	3 * T_{ADC_CLK}
0011	4 * T_{ADC_CLK}	4 * T_{ADC_CLK}	4 * T_{ADC_CLK}	4 * T_{ADC_CLK}
0100	5 * T_{ADC_CLK}	5 * T_{ADC_CLK}	5 * T_{ADC_CLK}	5 * T_{ADC_CLK}
0101	6 * T_{ADC_CLK}	6 * T_{ADC_CLK}	6 * T_{ADC_CLK}	6 * T_{ADC_CLK}
0110	7 * T_{ADC_CLK}	7 * T_{ADC_CLK}	7 * T_{ADC_CLK}	6 * T_{ADC_CLK}
0111	8 * T_{ADC_CLK}	8 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
1000	9 * T_{ADC_CLK}	9 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
1001	10 * T_{ADC_CLK}	10 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
1010	11 * T_{ADC_CLK}	10 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
1011	12 * T_{ADC_CLK}	10 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
others	12 * T_{ADC_CLK}	10 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}

21.7.3 ADC common regular data register for dual mode (ADC_CDR)

Address offset: 0x0C (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

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 21.4.31: 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 21.4.31: 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 172. ADC global register map(1)

Offset	Register
0x000 - 0x0B4	Master ADC1
0x0B8 - 0x0FC	Reserved
0x100 - 0x1B4	Slave ADC2
0x1B8 - 0x2FC	Reserved
0x300 - 0x30C	Master and slave ADCs common registers

1. Reserved area highlighted in gray.

Table 173. 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	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	ALIGN	RES [1:0]		DFSDMCFG	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
0x0C	**ADC_CFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x14	**ADC_SMPR1**	Res.	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
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.	Res.	Res.	Res.	LT1[11:0]												
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADC_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	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	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
0x44-0x48	Reserved	Res.																															

Table 173. 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
0x4C	**ADC_JSQR**	Res.	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]					Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL [3:0]				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]					Res.	Res.	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
0x64	**ADC_OFR2**	OFFSET2_EN	OFFSET2_CH[4:0]					Res.	Res.	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
0x68	**ADC_OFR3**	OFFSET3_EN	OFFSET3_CH[4:0]					Res.	Res.	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
0x6C	**ADC_OFR4**	OFFSET4_EN	OFFSET4_CH[4:0]					Res.	Res.	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
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.
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

Table 173. 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
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

Table 174. 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	**ADC_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	**ADC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH18SEL	CH17SEL	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	**ADC_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 2.3 on page 87* 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 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 161: Dual-channel DAC block diagram*)

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- 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 161 shows the block diagram of a DAC channel and *Table 176* gives the pin description.

22.3 DAC implementation

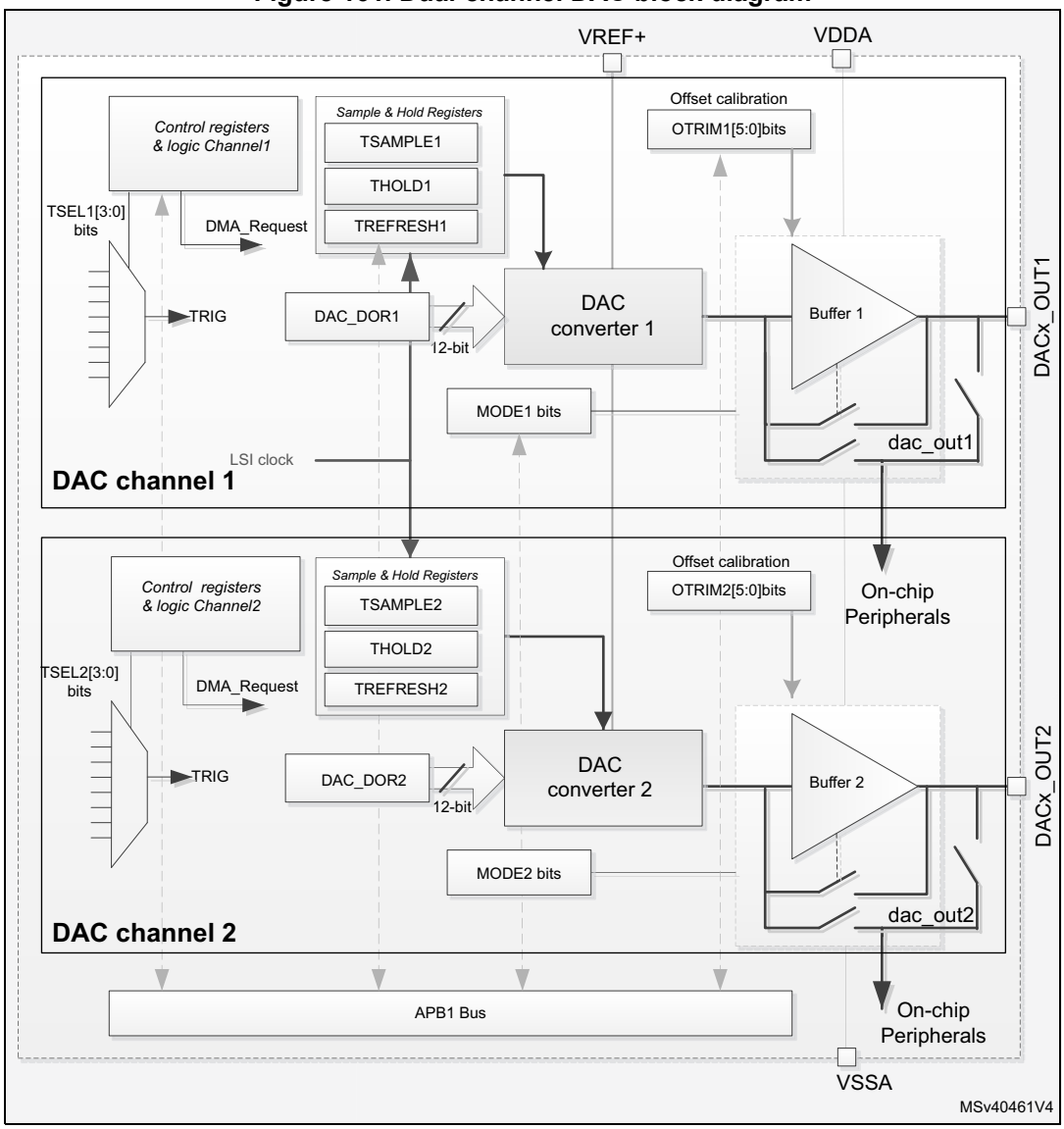
Table 175. DAC features

DAC features	DAC1
Dual channel	X
Output buffer	X
I/O connection	DAC1_OUT1 on PA4, DAC1_OUT2 on PA5
Maximum sampling time	1MSPS
Autonomous mode	-

22.4 DAC functional description

22.4.1 DAC block diagram

Figure 161. 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.

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 clock source 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 clock source.

Table 176. 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

22.4.2 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.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

22.4.3 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)

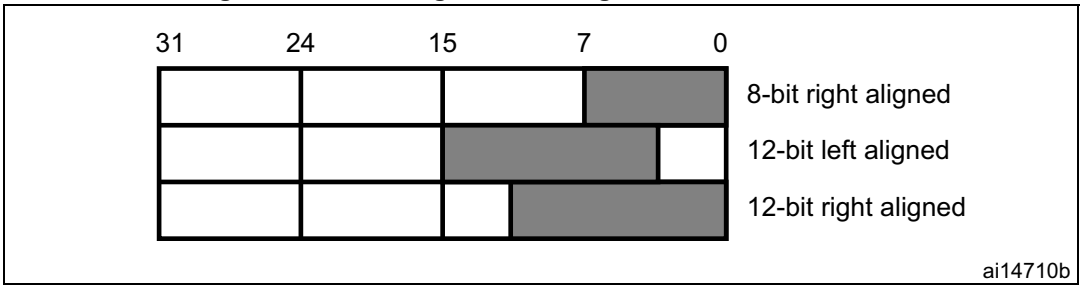
  – 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)

  – 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 162. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

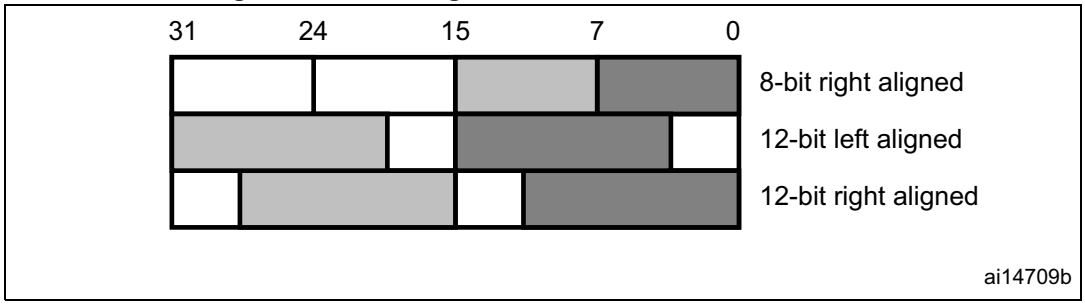
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 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 163. Data registers in dual DAC channel mode



22.4.4 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 APB1 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 APB1 clock cycles after the trigger signal.

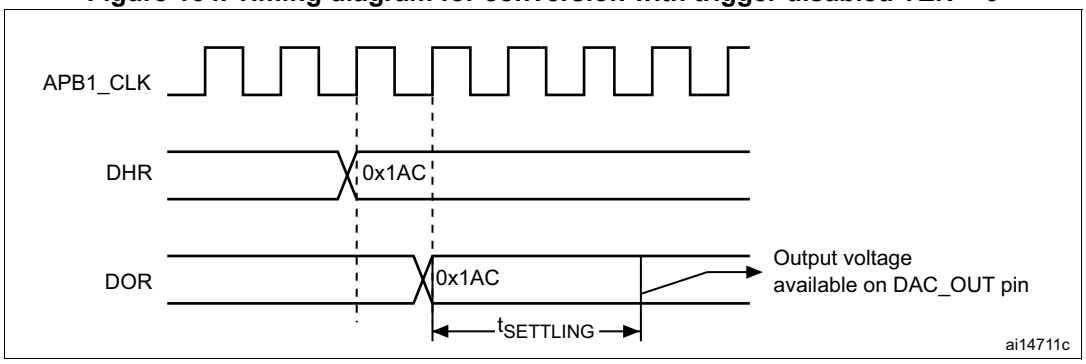
When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

HFSEL bit of DAC_CR must be set when APB1 clock speed is faster than 80 MHz. It adds an extra delay of three APB1 clock cycles to the transfer from DAC_DHRx register to DAC_DORx register ($t_{SETTLING}$).

The DAC_DORx update rate is limited to 1/3 of APB1 clock frequency. When HFSEL bit is set, this rate is limited to 1/8 of the APB1 clock frequency.

When HFSEL is set, it is not allowed to write the DHRx register during a period of eight clock cycles after the ENx bit is set. During this period, making software/hardware triggering is not allowed either.

Figure 164. Timing diagram for conversion with trigger disabled TEN = 0



22.4.5 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:

$$\text{DACoutput} = V_{REF} \times \frac{DOR}{4096}$$

22.4.6 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 *Table 177: DAC trigger selection*.

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 APB1 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.

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 APB1 clock cycle.

Table 177. DAC trigger selection

Source	Type	TSELx[3:0]
SWTRIG	Software control bit	0000
TIM1_TRGO	Internal signal from on-chip timers	0001
TIM2_TRGO	Internal signal from on-chip timers	0010
TIM4_TRGO	Internal signal from on-chip timers	0011
TIM5_TRGO	Internal signal from on-chip timers	0100
TIM6_TRGO	Internal signal from on-chip timers	0101
TIM7_TRGO	Internal signal from on-chip timers	0110
TIM8_TRGO	Internal signal from on-chip timers	0111
TIM15_TRGO	Internal signal from on-chip timers	1000
Reserved	-	1001
Reserved	-	1010
LPTIM1_OUT	Internal signal from on-chip timers	1011
LPTIM2_OUT	Internal signal from on-chip timers	1100
EXTI9	External pin	1101
Reserved	-	1110
Reserved	-	1111

22.4.7 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.

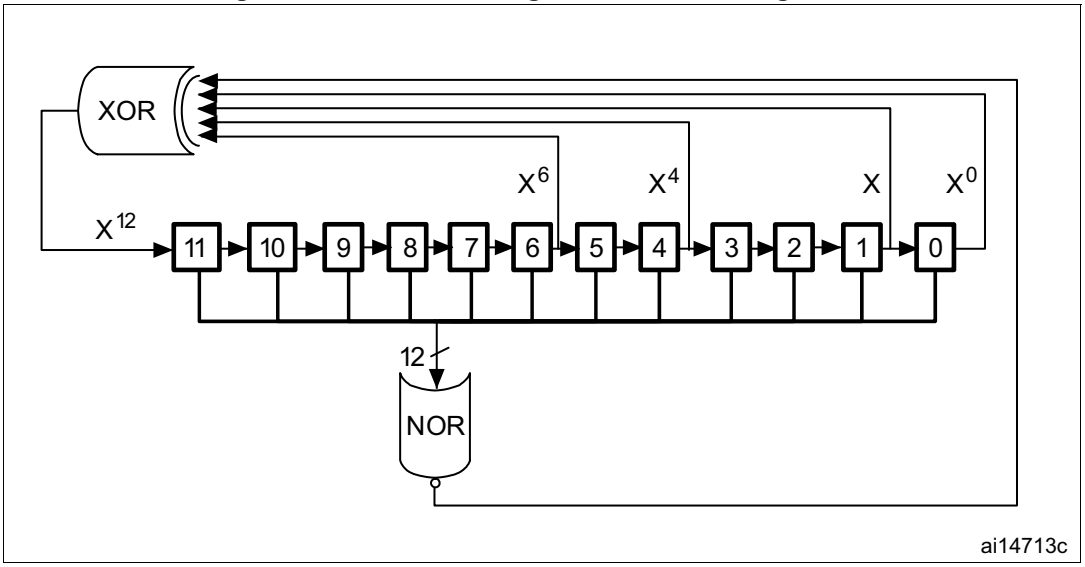
The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, 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.

22.4.8 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 APB1 clock cycles after each trigger event, following a specific calculation algorithm.

Figure 165. DAC LFSR register calculation algorithm

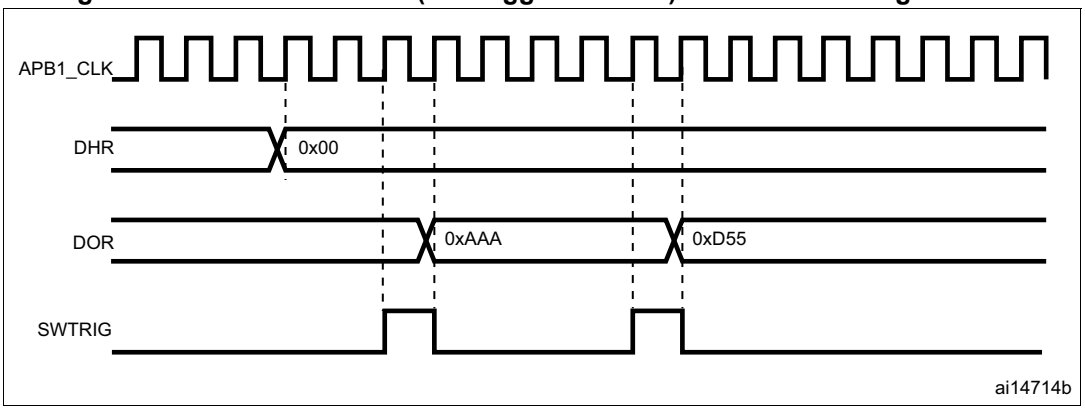


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

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 166. 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.

22.4.9 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 APB1 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 167. DAC triangle wave generation

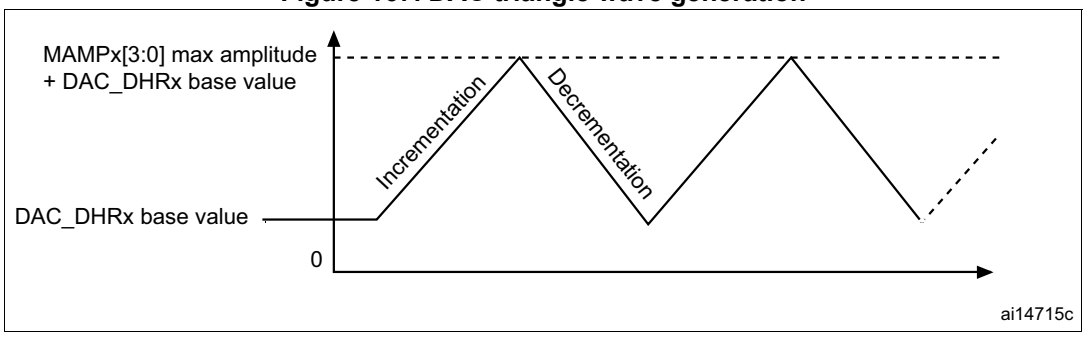
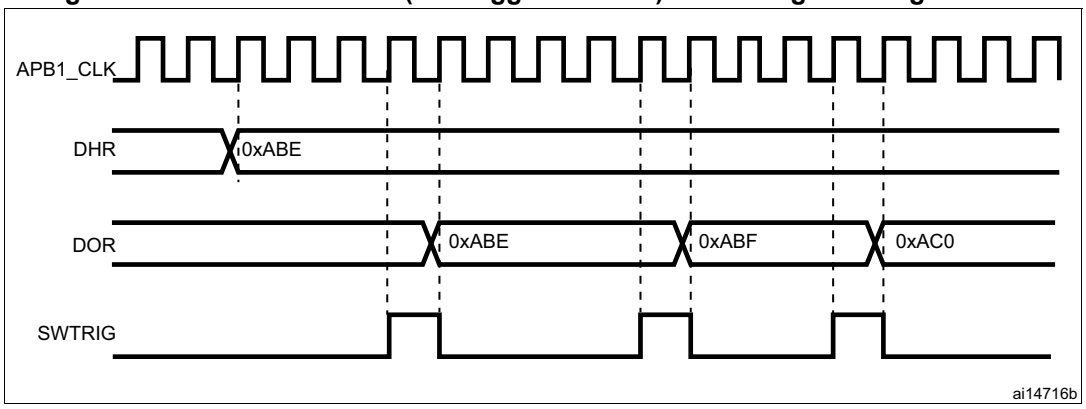


Figure 168. DAC conversion (SW trigger enabled) with triangle wave generation



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.

22.4.10 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 low-speed clock in addition to the APB1 clock, allowing to use the DAC channels in deep low power modes such as Stop mode.

The LSI low-speed clock 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 (APB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the 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 clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI ~32 KHz is selected:

12 cycles are required for sample phase: 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 178. 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 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms

(where R_{BON} = 2 kΩ)

Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs

(where N_{LSB} = 10 (10 LSB drop during the hold phase)

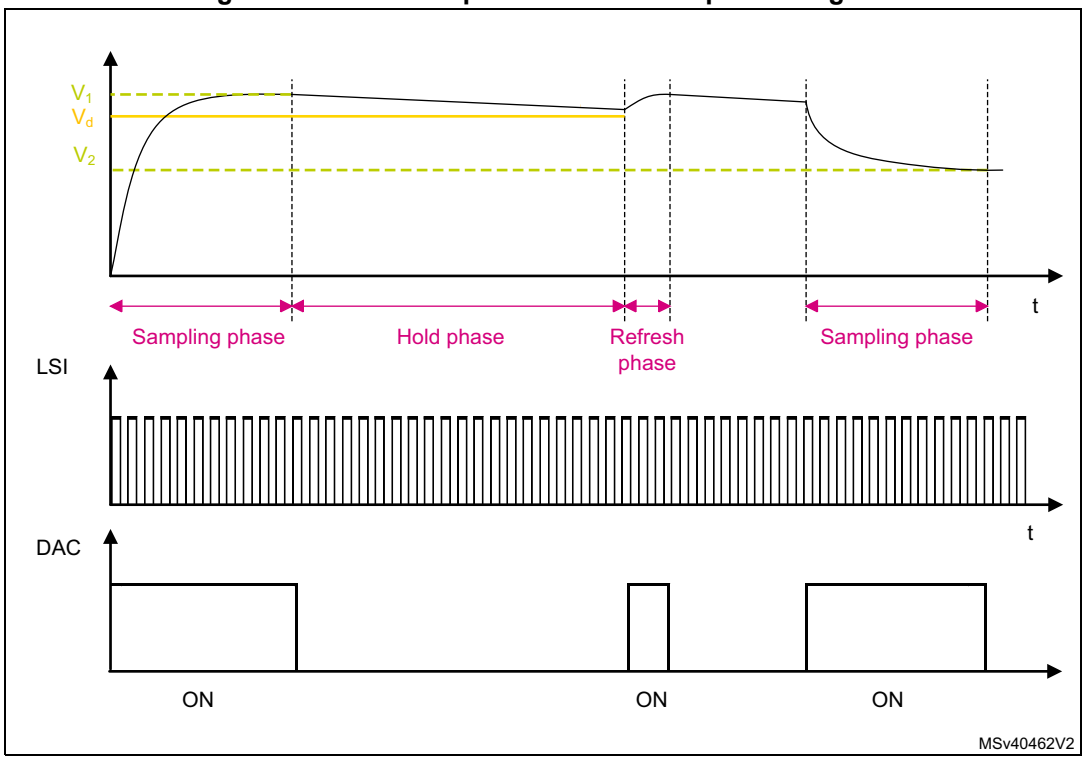
Hold phase:

D_v = i_{leak} * t_{hold} / C_{SH} = 0.0073 V (10 LSB of 12bit at 3 V)

i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)

t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 169. 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.

Table 179. 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)

Table 179. Channel output modes summary (continued)

MODEx[2:0]			Mode	Buffer	Output connections
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

22.4.11 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = ((D/2^{N-1}) \times G \times V_{ref}) + V_{OS}$$

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.

22.4.12 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.

11 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 APB1 clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three APB1 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 APB1 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 APB1 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 APB1 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 APB1 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 APB1 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 APB1 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 APB1 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 APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

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 APB1 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 APB1 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 APB1 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 APB1 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 APB1 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 APB1 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 APB1 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 APB1 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 APB 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 APB1 clock cycles later). Then the DAC channel2 triangle counter is updated.

22.5 DAC low-power modes

Table 180. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA
Low-power run	No effect.
Low-power sleep	No effect. DAC used with DMA.
Stop 0 / Stop 1	DAC remains active with a static value, if Sample and hold mode is selected using LSI clock
Stop 2	The DAC registers content is kept. The DAC must be disabled before entering Stop 2.

Table 180. Effect of low-power modes on DAC (continued)

Mode	Description
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

22.6 DAC interrupts

Table 181. 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 76* 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
HFSEL	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	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
1x: Triangle 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 22.4.6: DAC trigger selection* 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 APB1 clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three APB1 clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one APB1 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 **HFSEL**: High frequency interface mode enable

This bit is set and cleared by software to enable/disable DAC interface high speed mode

This bit need to be set when APB1 clock frequency is higher than 80 MHz.

0: High frequency interface mode disabled

1: High frequency interface mode enabled

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1=0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in Normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: wave generation disabled

01: Noise wave generation enabled

1x: Triangle wave generation enabled

Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trg1
0010: dac_ch1_trg2
...
1111: dac_ch1_trg15
Refer to the trigger selection tables in *Section 22.4.6: DAC trigger selection* 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 APB1 clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three APB1 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 APB1 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.	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.	SWTRIG2	SWTRIG1
														w	w

Bits 31:2 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 APB1 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 APB1 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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 31:12 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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 31:12 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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_ FLAG1	DMAU DR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													

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 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 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after Sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bits 10:0 Reserved, must be kept at reset value.

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE2[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.	MODE1[2:0]		
													rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

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

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

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

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1=0 and bit CEN1 =0 in the DAC_CR register). If EN1=1 or CEN1 =1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel1 mode:

– DAC channel1 in Normal mode

000: DAC channel1 is connected to external pin with Buffer enabled

001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled

010: DAC channel1 is connected to external pin with Buffer disabled

011: DAC channel1 is connected to on chip peripherals with Buffer disabled

– DAC channel1 in sample & hold mode

100: DAC channel1 is connected to external pin with Buffer enabled

101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel1 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN1=0.

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 clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI 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 clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI 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 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 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 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 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 register map

Table 182 summarizes the DAC registers.

Table 182. 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	HFSEL	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	0
0x04	DAC_SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	DAC_DHR12R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_DHR12L1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	DAC_DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	DAC_DHR12R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	DAC_DHR12L2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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				
0x1C	DAC_DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
	Reset value																									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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x34	DAC_SR	BWST2	CAL_FLAG2	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0	0	0													

Table 182. 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
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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value														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

Refer to *Section 2.3 on page 87* 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 two voltages[a], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 0: V_{REF_OUT1} around 2.048 V.
- VRS = 1: V_{REF_OUT2} around 2.5 V.

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 183. 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.

23.3 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. For the VRS = 1 setting, the calibration data is automatically loaded to the TRIM register at reset. For the VRS = 0

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

setting, the software must take care of copying the calibration data from the read-only system memory area (Flash memory) to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits.

Table 184. VREFBUF trimming data

Calibration value name	Description	Memory address
VREF_SC0	VREFBUF trimming value for VRS = 0	0x0BFA 0579
VREF_SC1	VREFBUF trimming value for VRS = 1	0x0BFA 0530

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.	Res.	Res.	VRR	VRS	HIZ	ENVR
												r	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **VRR:** Voltage reference buffer ready
0: the voltage reference buffer output is not ready.
1: the voltage reference buffer output reached the requested level.

Bit 2 **VRS**: Voltage reference scale
This bit selects the value generated by the voltage reference buffer.
0: Voltage reference set to V_{REF_OUT1} (around 2.048 V).
1: Voltage reference set to V_{REF_OUT2} (around 2.5 V).

Bit 1 **HIZ:** High impedance mode
This bit controls the analog switch to connect or not the V_{REF+} pin.
0: V_{REF+} pin is internally connected to the voltage reference buffer output.
1: V_{REF+} pin is high impedance.
Refer to *Table 183: 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

For VRS = 1 (2.5 V), these bits are automatically initialized after reset with the trimming value stored in the Flash memory during the production test.
For VRS = 0 (2.048V), the software must take care of copying the calibration data from the read-only system memory area (Flash memory) to the TRIM[5:0] bitfield. Writing into these bits allows the tuning of the internal reference buffer voltage.

Note: If the user application performs the trimming, the trimming code should 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 185. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**VREFBUF_CSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRR	VRS	HIZ	ENVR
	Reset value																													0	0	1	0
0x04	**VREFBUF_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

Refer to *Section 2.3 on page 87* for the register boundary addresses.

24 Comparator (COMP)

24.1 Introduction

The device embeds two ultra-low-power comparators COMP1, and COMP2

The comparators can be used for a variety of functions including:

- Wakeup 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

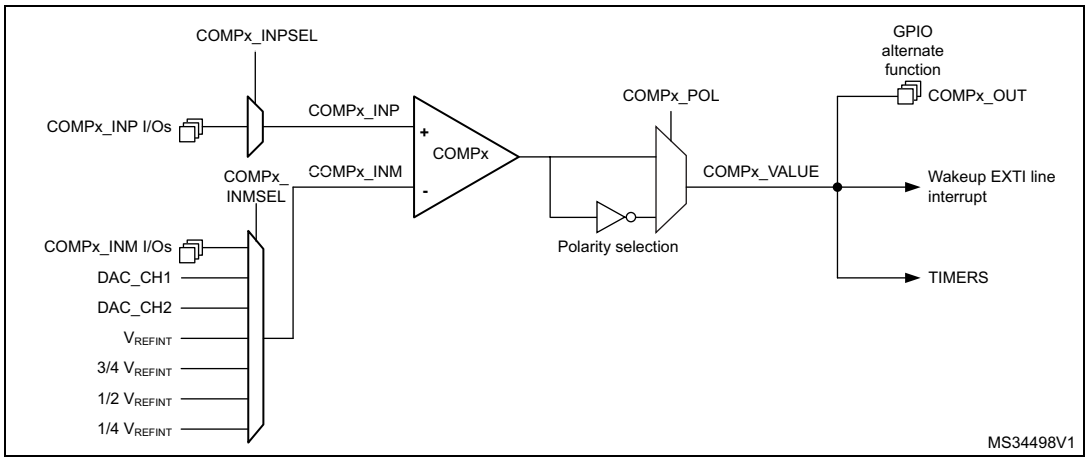
- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - DAC Channel1 and Channel2
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed / consumption
- The outputs can be redirected to an I/O or to timer inputs for triggering:
  - Break events for fast PWM shutdowns
- Comparator outputs with blanking source
- The two comparators can be combined in a window comparator
- Each comparator has interrupt generation capability with wakeup from Sleep and Stop modes (through the EXTI controller)

24.3 COMP functional description

24.3.1 COMP block diagram

The block diagram of the comparators is shown in *Figure 170*.

Figure 170. 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 186. COMP1 input plus assignment

COMP1_INP	COMP1_INPSEL [1:0]
PC5	00
PB2	01
PA2	10

Table 187. COMP1 input minus assignment

COMP1_INM	COMP1_INMSEL[2:0]
¼ V_{REFINT}	000
½ V_{REFINT}	001
¾ V_{REFINT}	010
V_{REFINT}	011
DAC Channel1	100
DAC Channel2	101
PB1	110
PC4	111

Table 188. COMP2 input plus assignment

COMP2_INP	COMP2_INPSEL
PB4	0
PB6	1

Table 189. COMP2 input minus assignment

COMP2_INM	COMP2_INMSEL[2:0]
¼ V_{REFINT}	000
½ V_{REFINT}	001
¾ V_{REFINT}	010
V_{REFINT}	011
DAC Channel1	100
DAC Channel2	101
PB3	110
PB7	111

24.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the APB2 clock.

There is no clock enable control bit provided in the RCC controller. Reset and clock enable bits are common for COMP and SYSCFG.

Important: The polarity selection logic and the output redirection to the port works independently from the APB2 clock. This allows the comparator to work even in Stop mode.

24.3.4 Comparator 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 to 1. This causes the whole register to become read-only, including the COMPx LOCK bit.

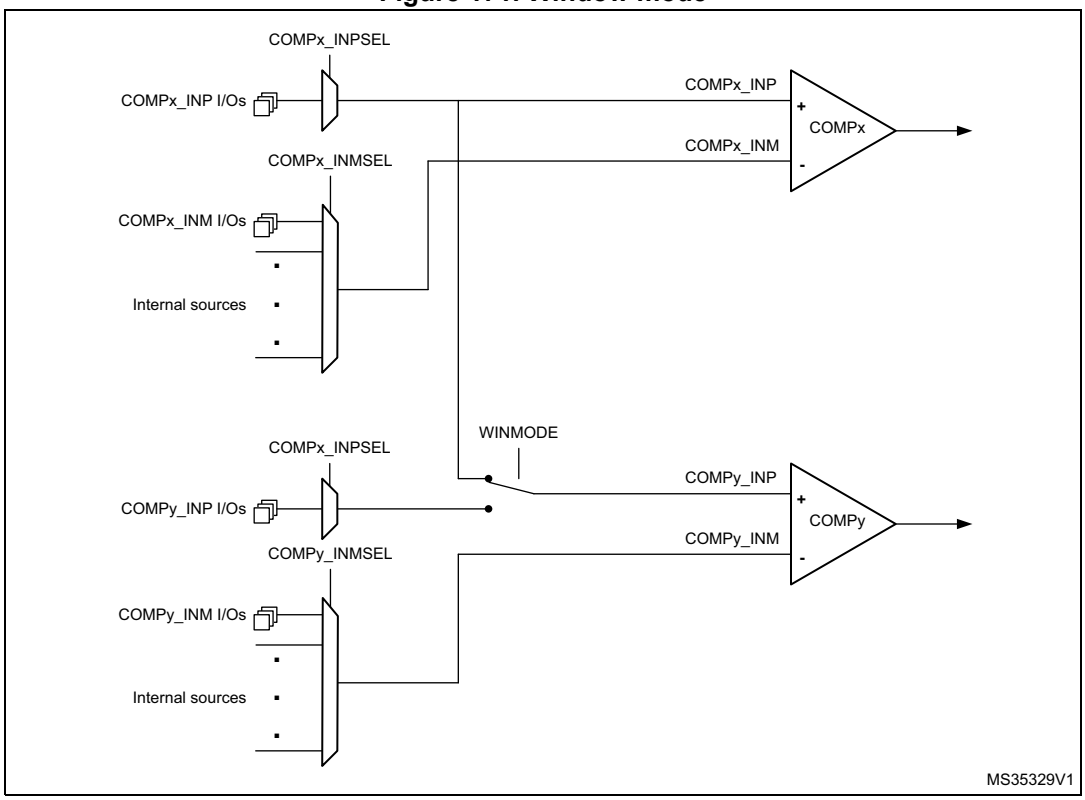
The write protection can only be reset by a MCU reset.

24.3.5 Window comparator

The purpose of window comparator is to monitor the analog voltage if it is within specified voltage range defined by lower and upper threshold.

Two embedded comparators can be utilized to create window comparator. The monitored analog voltage is connected to the non-inverting (plus) inputs of comparators connected together and the upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators. Two non-inverting inputs can be connected internally together by enabling WINMODE bit to save one IO for other purposes.

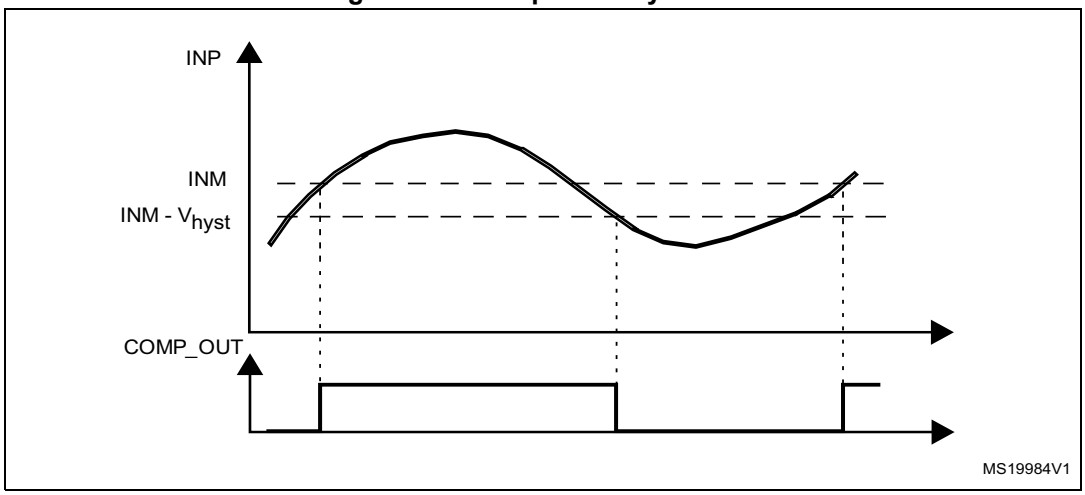
Figure 171. Window mode



24.3.6 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

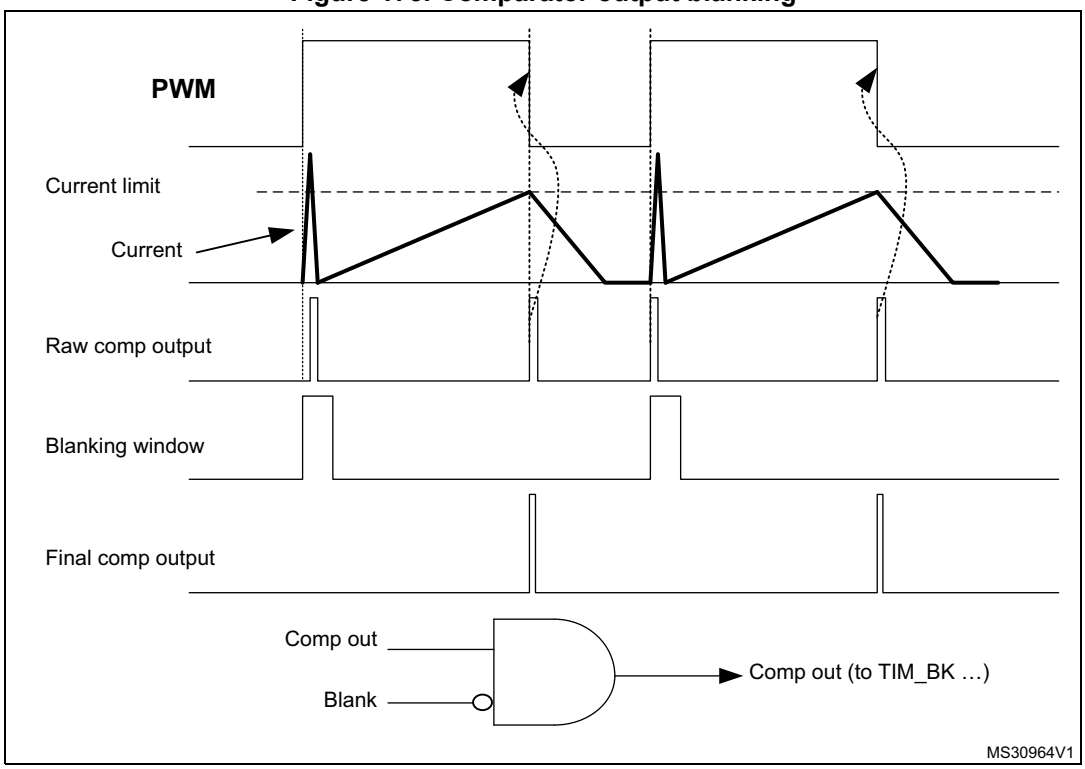
Figure 172. Comparator hysteresis



24.3.7 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes).It consists of a selection of a blanking window which is a timer output compare signal. The selection is done by software (refer to the comparator register description for possible blanking signals). Then, the complementary of the blanking signal is ANDed with the comparator output to provide the wanted comparator output. See the example provided in the figure below.

Figure 173. Comparator output blanking



24.3.8 COMP power and speed modes

COMP1 and COMP2 power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The bits PWRMODE[1:0] in COMPx_CSR registers can be programmed as follows:

00: High speed / full power

01 or 10: Medium speed / medium power

11: Low speed / ultra-low-power

24.4 COMP low-power modes

Table 190. Comparator behavior in the low power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Low-power run	No effect.
Low-power sleep	No effect. COMP interrupts cause the device to exit the Low-power sleep mode.
Stop 0	No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode.
Stop 1	
Stop 2	
Standby	The COMP registers are powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

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 from low-power modes.

Refer to Interrupt and events section for more details.

To enable 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 the rising, falling or both edges sensitivity
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines
3. Enable COMPx.

Table 191. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop modes	Exit from Standby mode
COMP1 output	VALUE in COMP1_CSR	Through EXTI	Yes	Yes	N/A
COMP2 output	VALUE in COMP2_CSR	Through EXTI	Yes	Yes	N/A

24.6 COMP registers

24.6.1 Comparator 1 control and status register (COMP1_CSR)

The COMP1_CSR is the Comparator 1 control/status register. It contains all the bits /flags related to comparator1.

Address offset: 0x00

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCAL EN	BRG EN	Res.	BLANKING			HYST	
rs	r							rw	rw		rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLA RITY	Res.	Res.	Res.	Res.	Res.	Res.	INP SEL.		INMSEL			PWRMODE		Res.	EN
rw							rw		rw			rw			rw

Bit 31 **LOCK:** COMP1_CSR register lock bit

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator 1 control register, COMP1_CSR[31:0].

0: COMP1_CSR[31:0] for comparator 1 are read/write

1: COMP1_CSR[31:0] for comparator 1 are read-only

Bit 30 **VALUE:** Comparator 1 output status bit

This bit is read-only. It reflects the current comparator 1 output taking into account POLARITY bit effect.

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

Bit 23 **SCALEN**: Voltage scaler enable bit

This bit is set and cleared by software. This bit enable the outputs of the V_{REFINT} divider available on the minus input of the Comparator 1.

0: Bandgap scaler disable (if SCALEN bit of COMP2_CSR register is also reset)

1: Bandgap scaler enable

Bit 22 **BRGEN:** Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enable the bridge of the scaler.

0: Scaler resistor bridge disable (if BRGEN bit of COMP2_CSR register is also reset)

1: Scaler resistor bridge enable

If SCALEN is set and BRGEN is reset, BG voltage reference is available but not 1/4 BGAP, 1/2 BGAP, 3/4 BGAP. BGAP value is sent instead of 1/4 BGAP, 1/2 BGAP, 3/4 BGAP.

If SCALEN and BRGEN are set, 1/4 BGAP 1/2 BGAP 3/4 BGAP and BGAP voltage references are available.

Bit 21 Reserved, must be kept at reset value

Bits 20:18 **BLANKING[2:0]:** Comparator 1 blanking source selection bits

These bits select which timer output controls the comparator 1 output blanking.

000: No blanking

001: TIM1 OC5 selected as blanking source

010: TIM2 OC3 selected as blanking source

100: TIM3 OC3 selected as blanking source

All other values: reserved

Bits 17:16 **HYST[1:0]:** Comparator 1 hysteresis selection bits

These bits are set and cleared by software (only if LOCK not set). They select the hysteresis voltage of the comparator 1.

00: No hysteresis

01: Low hysteresis

10: Medium hysteresis

11: High hysteresis

Bit 15 **POLARITY:** Comparator 1 polarity selection bit

This bit is set and cleared by software (only if LOCK not set). It inverts Comparator 1 polarity.

0: Comparator 1 output value not inverted

1: Comparator 1 output value inverted

Bits 14:9 Reserved, must be kept at reset value.

Bits 8:7 **INPSEL**: Comparator1 input plus selection bit

This bit is set and cleared by software (only if LOCK not set).

00: PC5

01: PB2

10: PA2

11: Reserved

Bits 6:4 **INMSEL**: Comparator 1 input minus selection bits

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of comparator 1.

000 = 1/4 V_{REFINT}

001 = 1/2 V_{REFINT}

010 = 3/4 V_{REFINT}

011 = V_{REFINT}

100 = DAC Channel1

101 = DAC Channel2

110 = PB1111 = PC4

Bits 3:2 **PWRMODE[1:0]:** Power Mode of the comparator 1

These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the Comparator 1.

00: High speed

01 or 10: Medium speed

11: Ultra low power

Bit 1 Reserved, must be kept cleared.

Bit 0 **EN:** Comparator 1 enable bit

This bit is set and cleared by software (only if LOCK not set). It switches on Comparator1.

0: Comparator 1 switched OFF

1: Comparator 1 switched ON

24.6.2 Comparator 2 control and status register (COMP2_CSR)

The COMP2_CSR is the Comparator 2 control/status register. It contains all the bits /flags related to comparator 2.

Address offset: 0x04

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCAL EN	BRG EN	Res.	BLANKING			HYST	
rs	r							rw	rw		rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLA RITY	Res.	Res.	Res.	Res.	Res.	WIN MODE	Res.	INP SEL	INMSEL			PWRMODE		Res.	EN
rw						rw		rw	rw			rw			rw

Bit 31 **LOCK:** CSR register lock bit

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator 2 control register, COMP2_CSR[31:0].

0: COMP2_CSR[31:0] for comparator 2 are read/write

1: COMP2_CSR[31:0] for comparator 2 are read-only

Bit 30 **VALUE:** Comparator 2 output status bit

This bit is read-only. It reflects the current comparator 2 output taking into account POLARITY bit effect.

Bits 29: Reserved, must be kept at reset value

Bit 23 **SCALEN**: Voltage scaler enable bit

This bit is set and cleared by software. This bit enable the outputs of the V_{REFINT} divider available on the minus input of the Comparator 2.

0: Bandgap scaler disable (if SCALEN bit of COMP1_CSR register is also reset)

1: Bandgap scaler enable

Bit 22 **BRGEN:** Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enable the bridge of the scaler.

0: Scaler resistor bridge disable (if BRGEN bit of COMP1_CSR register is also reset)

1: Scaler resistor bridge enable

If SCALEN is set and BRGEN is reset, BG voltage reference is available but not 1/4 BGAP, 1/2 BGAP, 3/4 BGAP. BGAP value is sent instead of 1/4 BGAP, 1/2 BGAP, 3/4 BGAP.

If SCALEN and BRGEN are set, 1/4 BGAP 1/2 BGAP 3/4 BGAP and BGAP voltage references are available.

Bit 21 Reserved, must be kept at reset value

Bits 20:18 **BLANKING[2:0]:** Comparator 2 blanking source selection bits

These bits select which timer output controls the comparator 2 output blanking.

000: No blanking

001: TIM3 OC4 selected as blanking source

010: TIM8 OC5 selected as blanking source

100: TIM15 OC1 selected as blanking source

All other values: reserved

Bits 17:16 **HYST[1:0]:** Comparator 2 hysteresis selection bits

These bits are set and cleared by software (only if LOCK not set). Select the hysteresis voltage of the comparator 2.

00: No hysteresis

01: Low hysteresis

10: Medium hysteresis

11: High hysteresis

Bit 15 **POLARITY:** Comparator 2 polarity selection bit

This bit is set and cleared by software (only if LOCK not set). It inverts Comparator 2 polarity.

0: Comparator 2 output value not inverted

1: Comparator 2 output value inverted

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **WINMODE**: Windows mode selection bit

This bit is set and cleared by software (only if LOCK not set). This bit selects the window mode of the comparators. If set, both positive inputs of comparators will be connected together.

0: Input plus of Comparator 2 is not connected to Comparator 1

1: Input plus of Comparator 2 is connected with input plus of Comparator 1

Bit 8 Reserved, must be kept at reset value.

Bit 7 **INPSEL**: Comparator 1 input plus selection bit

This bit is set and cleared by software (only if LOCK not set).

0: PB4

1: PB6

Bits 6:4 **INMSEL**: Comparator 2 input minus selection bits

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of comparator 2.

000 = 1/4 V_{REFINT}
001 = 1/2 V_{REFINT}
010 = 3/4 V_{REFINT}
011 = V_{REFINT}
100 = DAC Channel1
101 = DAC Channel2
110 = PB3
111 = PB7

Bits 3:2 **PWRMODE[1:0]:** Power Mode of the comparator 2

These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the Comparator 2.

00: High speed
01 or 10: Medium speed
11: Ultra low power

Bit 1 Reserved, must be kept cleared.

Bit 0 **EN:** Comparator 2 enable bit

This bit is set and cleared by software (only if LOCK not set). It switches oncomparator2.

0: Comparator 2 switched OFF
1: Comparator 2 switched ON

24.6.3 COMP register map

The following table summarizes the comparator registers.

Table 192. 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	**COMP1_CSR**	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN.	Res.	BLANKING			HYST		POLARITY.	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL.		INMSEL			PWRMODE		Res.	EN
	Reset value	0	0							0	0		0	0	0	0	0	0							0	0	0	0	0	0	0		0
0x04	**COMP2_CSR**	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	Res.	BLANKING			HYST		POLARITY.	Res.	Res.	Res.	Res.	Res.	WINMODE	Res.	INPSEL	INMSEL			PWRMODE		Res.	EN
	Reset value	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 87* for the register boundary addresses.

25 Operational amplifiers (OPAMP)

25.1 Introduction

The device embeds two operational amplifiers with two inputs and one output each. The three I/Os can be connected to the external pins, this enables any type of external interconnections. The operational amplifier can be configured internally as a follower or as an amplifier with a non-inverting gain ranging from 2 to 16.

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 (down to 1 nA)
- Low input offset voltage (1.5 mV after calibration, 3 mV with factory calibration)
- Low-power mode (current consumption reduced to 30 µA instead of 100 µA)
- Fast wakeup time (10 µs in normal mode, 30 µs in low-power mode)
- Gain bandwidth of 1.6 MHz

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.

There are two functional modes, the low-power mode or the normal mode. In functional mode the inputs and output of the OPAMP are connected as described in the *Section 25.3.3: Signal routing*.

25.3.1 OPAMP reset and clocks

The operational amplifier clock is necessary for accessing the registers. When the application does not need to have read or write access to those registers, the clock can be switched off using the peripheral clock enable register (see OPAMPEN bit in *Section 9.8.19: RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1)*).

The bit OPAEN enables and disables the OPAMP operation. The OPAMP registers configurations should 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.

25.3.2 Initial configuration

The default configuration of the operational amplifier is a functional mode where the three IOs are connected to external pins. In the default mode the operational amplifier uses the factory trimming values. See electrical characteristics section of the datasheet for factory trimming conditions, usually the temperature is 30 °C and the voltage is 3 V. The trimming values can be adjusted, see *Section 25.3.5: Calibration* for changing the trimming values. The default configuration uses the normal mode, which provides the highest performance. Bit OPALPM can be set in order to switch the operational amplifier to low-power mode and reduced performance. Both normal and low-power mode characteristics are defined in the section “electrical characteristics” of the datasheet. Before utilization, the bit OPA_RANGE of OPAMP_CSR must be set to 1 if V_{DDA} is above 2.4V, or kept at 0 otherwise.

As soon as the OPAEN bit in OPAMP_CSR register is set, the operational amplifier is functional. The two input pins and the output pin are connected as defined in *Section 25.3.3: 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.

25.3.3 Signal routing

The routing for the operational amplifier pins is determined by OPAMP_CSR register.

The connections of the two operational amplifiers (OPAMP1 and OPAMP2) are described in the table below

Table 193. Operational amplifier possible connections

Signal	Pin	Internal	comment
OPAMP1_VINM	PA1 or dedicated pin(1)	OPAMP1_OUT or PGA	Controlled by bits OPAMODE and VM_SEL.
OPAMP1_VINP	PA0	DAC1_OUT1	Controlled by bit VP_SEL.
OPAMP1_VOUT	PA3	ADC1_IN8	The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC.
OPAMP2_VINM	PA7 or dedicated pin*Chapter 1.*	OPAMP2_OUT or PGA	Controlled by bits OPAMODE and VM_SEL.
OPAMP2_VINP	PA6	DAC1_OUT2	Controlled by bit VP_SEL
OPAMP2_VOUT	PB0	ADC1_IN15	The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC.

1. The dedicated pin is only available on BGA132 and BGA169 package. This configuration provides the lowest input bias current (see datasheet).

25.3.4 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 amplifier output pin is directly connected to the output pad to minimize the output impedance. It cannot be used as a general purpose I/O, even if the amplifier is configured as a PGA and only connected to the ADC channel.

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)

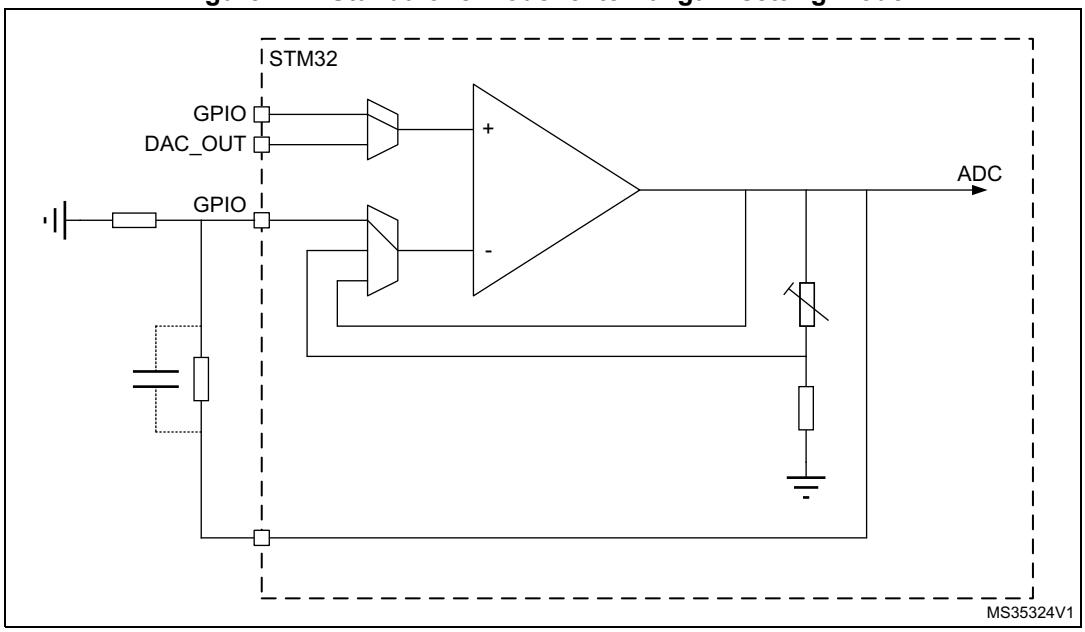
The procedure to use the OPAMP in standalone mode is presented hereafter.

Starting from the default value of OPAMP_CSR, and the default state of GPIOx_MODER, configure bit OPA_RANGE according the V_{DDA} voltage. 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:

- OPALPM can be set to “operational amplifier low-power” mode in order to save power.
- USERTRIM can be set to modify the trimming values for the input offset.

Figure 174. Standalone mode: external gain setting mode



Follower configuration mode

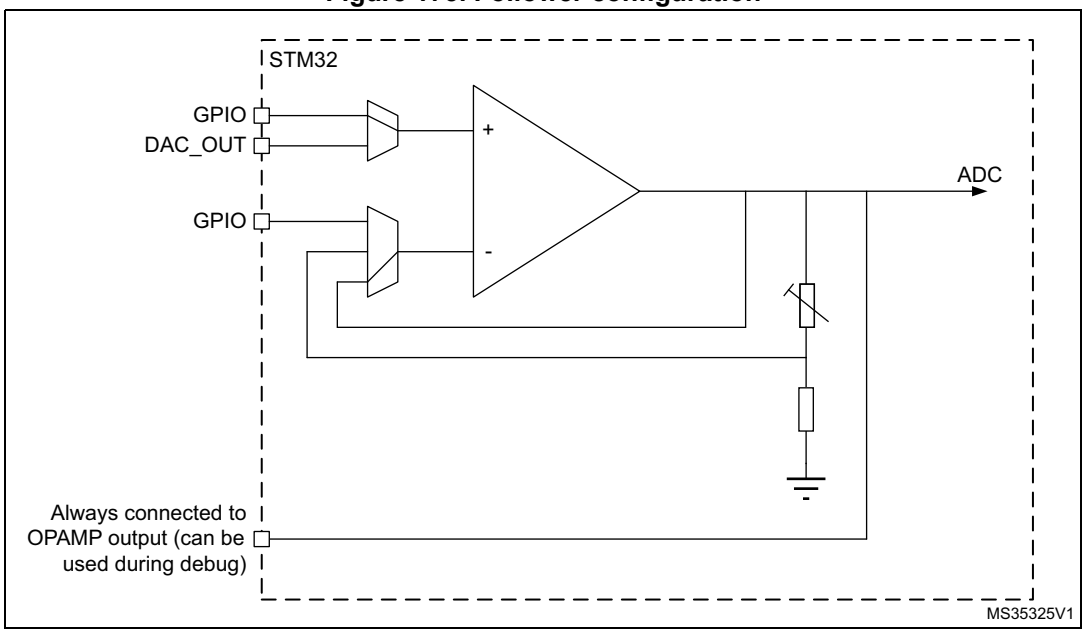
The procedure to use the OPAMP in follower mode is presented hereafter.

- configure OPAMODE bits as “internal follower”
- configure VP_SEL bits as “GPIO connected to VINP”.
- As soon as the OPAEN bit is set, the signal on pin OPAMP_VINP is copied to pin OPAMP_VOUT.

Note: The pin corresponding to OPAMP_VINM is free for another usage.

Note: The signal on the operational amplifier 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 175. Follower configuration



Programmable Gain Amplifier mode

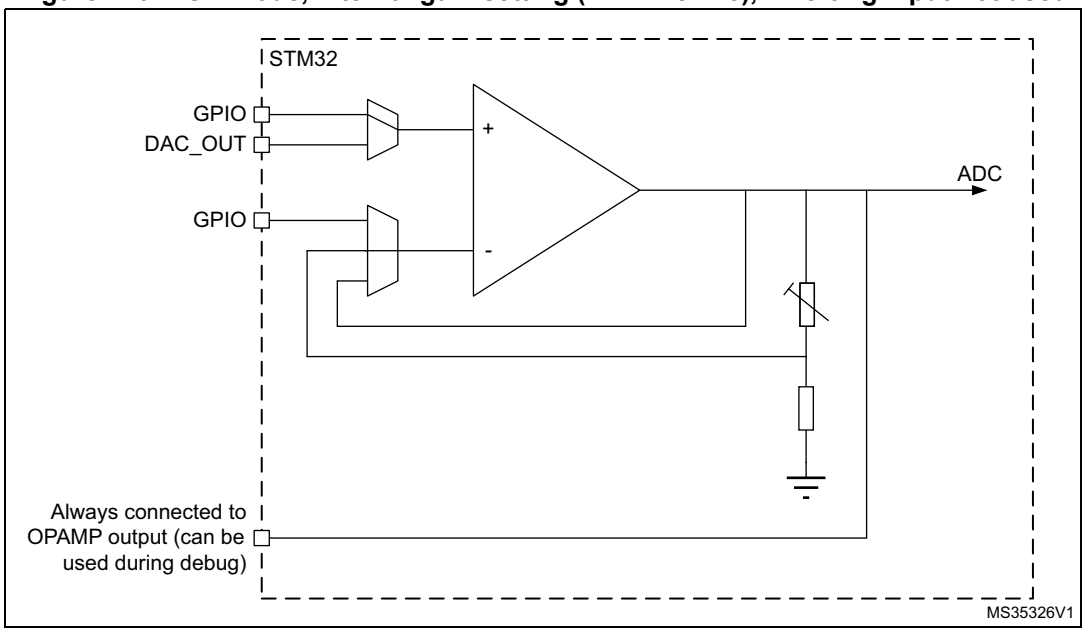
The procedure to use the OPAMP to amplify the amplitude of an input signal is presented hereafter.

- configure OPAMODE bits as “internal PGA enabled”,
- configure PGA_GAIN bits as “internal PGA Gain 2, 4, 8 or 16”,
- configure VM_SEL bits as “inverting not externally connected”,
- configure VP_SEL bits as “GPIO connected to VINP”.

As soon as the OPAEN bit is set, the signal on pin OPAMP_VINP is amplified by the selected gain and visible on pin OPAMP_VOUT.

Note: To avoid saturation, the input voltage should stay below V_{DDA} divided by the selected gain.

Figure 176. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used



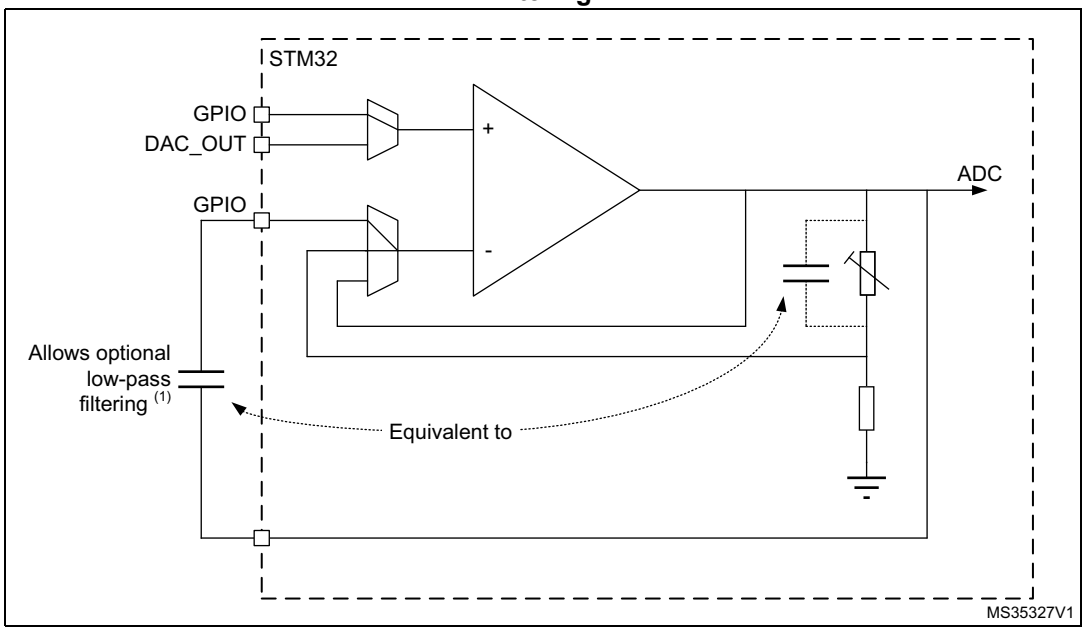
Programmable Gain Amplifier mode with external filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal, with an external filtering, is presented hereafter.

- configure OPAMODE bits as “internal PGA enabled”,
- configure PGA_GAIN bits as “internal PGA Gain 2, 4, 8 or 16”,
- configure VM_SEL bits as “GPIO connected to VINM”,
- configure VP_SEL bits as “GPIO connected to VINP”.

Any external connection on VINP can be used in parallel with the internal PGA, for example a capacitor can be connected between VOUT and VINM for filtering purpose (see datasheet for the value of resistors used in the PGA resistor network).

Figure 177. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering



1. The gain depends on the cut-off frequency.

25.3.5 Calibration

At startup, the trimming values are initialized with the preset ‘factory’ trimming value.

Each operational amplifier offset can be trimmed by the user. Specific registers allow to have different trimming values for normal mode and for low-power 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 inputs offset voltage to less than +/-1.5 mV within stable voltage and temperature conditions.

For each operational amplifier and each mode two trimming values need to be trimmed, one for N differential pair and one for P differential pair.

There are two registers for trimming the offsets for each operational amplifiers, one for normal mode (OPAMP_OTR) and one low-power mode (OPAMP_LPOTR). Each register is composed of five bits for P differential pair trimming and five bits for N differential pair trimming. These are the ‘user’ values.

The user is able to switch from ‘factory’ values to ‘user’ trimmed values using the USERTRIM bit in the OPAMP_CSR register. This bit is reset at startup and so the ‘factory’ value are applied by default to the OPAMP trimming registers.

User is liable to change the trimming values in calibration or in functional mode.

The offset trimming registers 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 functional environment.

- Setting CALSEL to 1 initializes the offset calibration for the P differential pair (low voltage reference used).
- Resetting CALSEL to 0 initializes the offset calibration for the N differential pair (high voltage reference used).

When CALON = 1, the bit CALOUT will reflect the influence of the trimming value selected by CALSEL and OPALPM. When the value of CALOUT switches between two consecutive trimming values, this means that those two values are the best trimming values. 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 194. Operating modes and calibration

Mode	Control bits				Output	
	OPAEN	OPALPM	CALON	CALSEL	V_{OUT}	CALOUT flag
Normal operating mode	1	0	0	X	analog	0
Low-power mode	1	1	0	X	analog	0
Power down	0	X	X	X	Z	0
Offset cal high for normal mode	1	0	1	0	analog	X
Offset cal low for normal mode	1	0	1	1	analog	X
Offset cal high for low-power mode	1	1	1	0	analog	X
Offset cal low for low-power mode	1	1	1	1	analog	X

Calibration procedure

Here are the steps to perform a full calibration of either one of the operational amplifiers:

1. Select correct OPA_RANGE in OPAMP_CSR, then set the OPAEN bit in OPAMP_CSR to 1 to enable the operational amplifier.
2. Set the USERTRIM bit in the OPAMP_CSR register to 1.
3. Choose a calibration mode (refer to *Table 194: Operating modes and calibration*). The steps 3 to 4 will have to be repeated 4 times. For the first iteration select
   – Normal mode, offset cal high (N differential pair)

   The above calibration mode correspond to OPALPM=0 and CALSEL=0 in the OPAMP_CSR register.
4. Increment TRIMOFFSETN[4:0] in OPAMP_OTR starting from 00000b until CALOUT changes to 1 in OPAMP_CSR.

Note: CALOUT will switch from 0 to 1 for offset cal high and from 1 to 0 for offset cal low.

Note: Between the write to the OPAMP_OTR register 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 offset is correctly compensated and that the corresponding trim code must be saved in the OPAMP_OTR register.

Repeat steps 3 to 4 for:

– Normal_mode and offset cal low
– Low power mode and offset cal high
– Low power mode and offset cal low

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.

During the calibration procedure, it is necessary to set up OPAMODE bits as 00 or 01 (PGA disable) or 11 (internal follower).

25.4 OPAMP low-power modes

Table 195. 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.
Stop 2	OPAMP registers content is kept. OPAMP must be disabled before entering Stop 2 mode.

Table 195. Effect of low-power modes on the OPAMP (continued)

Mode	Description
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
OPA_ RANGE	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
CAL OUT	USER TRIM	CAL SEL	CALON	Res.	VP_ SEL	VM_SEL		Res.	Res.	PGA_GAIN		OPAMODE		OPA LPM	OPAEN
r	rw	rw	rw		rw	rw	rw			rw	rw	rw	w	rw	rw

Bit 31 **OPA_RANGE:** Operational amplifier power supply range for stability
All AOP must be in power down to allow AOP-RANGE bit write. It applies to all AOP embedded in the product.
0: Low range (VDDA < 2.4V)
1: High range (VDDA > 2.4V)

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

Bit 15 **CALOUT:** Operational amplifier calibration output
During calibration mode offset is trimmed when this signal toggle.

Bit 14 **USERTRIM:** allows to switch from ‘factory’ AOP offset trimmed values to AOP offset ‘user’ trimmed values
This bit is active for both mode normal and low-power.
0: ‘factory’ trim code used
1: ‘user’ trim code used

Bit 13 **CALSEL:** Calibration selection
0: NMOS calibration (200mV applied on OPAMP inputs)
1: PMOS calibration (VDDA-200mV applied on OPAMP inputs)

Bit 12 **CALON:** Calibration mode enabled
0: Normal mode
1: Calibration mode (all switches opened by HW)

Bit 11 Reserved, must be kept at reset value.

Bit 10 **VP_SEL:** Non inverted input selection
0: GPIO connected to VINP
1: DAC connected to VINP

Bits 9:8 **VM_SEL:** Inverting input selection
These bits are used only when OPAMODE = 00, 01 or 10.
00: GPIO connected to VINM (valid also in PGA mode for filtering
01: Dedicated low leakage input, connected to VINM (valid also in PGA mode for filtering)
1x: Inverting input not externally connected. These configurations are valid only when OPAMODE = 10 (PGA mode)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PGA_GAIN:** Operational amplifier Programmable amplifier gain value
00: internal PGA Gain 2
01: internal PGA Gain 4
10: internal PGA Gain 8
11: internal PGA Gain 16

Bits 3:2 **OPAMODE:** Operational amplifier PGA mode
00: internal PGA disable
01: internal PGA disable
10: internal PGA enable, gain programmed in PGA_GAIN
11: internal follower

Bit 1 **OPALPM:** Operational amplifier Low Power Mode
The operational amplifier must be disable to change this configuration.
0: operational amplifier in normal mode
1: operational amplifier in low-power mode

Bit 0 **OPAEN:** Operational amplifier Enable
0: operational amplifier disabled
1: operational amplifier enabled

25.5.2 OPAMP1 offset trimming register in normal mode (OPAMP1_OTR)

Address offset: 0x04

Reset value: 0x0000 XXXX (factory trimmed values)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMOFFSETP[4:0]					Res.	Res.	Res.	TRIMOFFSETN[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 **TRIMOFFSETP[4:0]:** Trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMOFFSETN[4:0]:** Trim for NMOS differential pairs

25.5.3 OPAMP1 offset trimming register in low-power mode (OPAMP1_LPOTR)

Address offset: 0x08

Reset value: 0x0000 XXXX (factory trimmed values)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMLPOFFSETP[4:0]					Res.	Res.	Res.	TRIMLPOFFSETN[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 **TRIMLPOFFSETP[4:0]:** Low-power mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMLPOFFSETN[4:0]:** Low-power mode trim for NMOS differential pairs

25.5.4 OPAMP2 control/status register (OPAMP2_CRS)

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
CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL		Res.	Res.	PGA_GAIN		OPAMODE		OPALPM	OPAEN
r	rw	rw	rw		rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALOUT**: Operational amplifier calibration output
During calibration mode offset is trimmed when this signal toggle.

Bit 14 **USERTRIM**: allows to switch from ‘factory’ AOP offset trimmed values to AOP offset ‘user’ trimmed values
This bit is active for both mode normal and low-power.
0: ‘factory’ trim code used
1: ‘user’ trim code used

Bit 13 **CALSEL**: Calibration selection
0: NMOS calibration (200mV applied on OPAMP inputs)
1: PMOS calibration (VDDA-200mV applied on OPAMP inputs)

Bit 12 **CALON**: Calibration mode enabled
0: Normal mode
1: Calibration mode (all switches opened by HW)

Bit 11 Reserved, must be kept at reset value.

Bit 10 **VP_SEL**: Non inverted input selection
0: GPIO connected to VINP
1: DAC connected to VINP

Bits 9:8 **VM_SEL:** Inverting input selection
These bits are used only when OPAMODE = 00, 01 or 10.
00:GPIO connected to VINM (valid also in PGA mode for filtering)
01: Dedicated low leakage input, (available only on BGA132) connected to VINM (valid also in PGA mode for filtering)
1x: Inverting input not externally connected. These configurations are valid only when OPAMODE = 10 (PGA mode)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PGA_GAIN**: Operational amplifier Programmable amplifier gain value
00: internal PGA Gain 2
01: internal PGA Gain 4
10: internal PGA Gain 8
11: internal PGA Gain 16

Bits 3:2 **OPAMODE**: Operational amplifier PGA mode
00: internal PGA disable
01: internal PGA disable
10: internal PGA enable, gain programmed in PGA_GAIN
11: internal follower

Bit 1 **OPALPM**: Operational amplifier Low Power Mode
The operational amplifier must be disable to change this configuration.
0: operational amplifier in normal mode
1: operational amplifier in low-power mode

Bit 0 **OPAEN**: Operational amplifier Enable
0: operational amplifier disabled
1: operational amplifier enabled

25.5.5 OPAMP2 offset trimming register in normal mode (OPAMP2_OTR)

Address offset: 0x14

Reset value: 0x0000 XXXX (factory trimmed values)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMOFFSETP[4:0]					Res.	Res.	Res.	TRIMOFFSETN[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 **TRIMOFFSETP[4:0]**: Trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMOFFSETN[4:0]**: Trim for NMOS differential pairs

25.5.6 OPAMP2 offset trimming register in low-power mode (OPAMP2_LPOTR)

Address offset: 0x18

Reset value: 0x0000 XXXX (factory trimmed values)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMLPOFFSETP[4:0]					Res.	Res.	Res.	TRIMLPOFFSETN[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 **TRIMLPOFFSETP[4:0]**: Low-power mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMLPOFFSETN[4:0]**: Low-power mode trim for NMOS differential pairs

25.5.7 OPAMP register map

Table 196. OPAMP 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	**OPAMP1_CSR**	OPA_RANGE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL		Res.	Res.	PGA_GAIN		OPAMODE		OPALPM	OPAEN
	Reset value	0																0	0	0	0		0	0	0			0	0	0	0	0	0
0x04	**OPAMP1_OTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM OFFSETP[4:0]					Res.	Res.	Res.	TRIM OFFSETN[4:0]				
	Reset value																				(1)								(1)				
0x08	**OPAMP1_ LPOTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMLP OFFSETP[4:0]					Res.	Res.	Res.	TRIMLP OFFSETN[4:0]				
	Reset value																				(1)								(1)				
0x10	**OPAMP2_CSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL		Res.	Res.	PGA_GAIN		OPAMODE		OPALPM	OPAEN
	Reset value																	0	0	0	0		0	0	0			0	0	0	0	0	0
0x14	**OPAMP2_OTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM OFFSETP[4:0]					Res.	Res.	Res.	TRIM OFFSETN[4:0]				
	Reset value																				(1)								(1)				
0x18	**OPAMP2_LPO TR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMLP OFFSETP[4:0]					Res.	Res.	Res.	TRIMLP OFFSETN[4:0]				
	Reset value																				(1)								(1)				

1. Factory trimmed values.

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the register boundary addresses.

26 Digital filter for sigma delta modulators (DFSDM)

26.1 Introduction

Digital filter for sigma delta modulators (DFSDM) is a high-performance module dedicated to interface external ΣΔ modulators. It is featuring up to 4 external digital serial interfaces (channels) and up to 4 digital filters with flexible Sigma Delta stream digital processing options to offer up to 24-bit final ADC resolution. DFSDM also features optional parallel data stream input from internal ADC peripherals or from device memory.

An external ΣΔ modulator provides digital data stream of converted analog values from the external ΣΔ modulator analog input. This digital data stream is sent into a DFSDM input channel through a serial interface. DFSDM supports several standards to connect various ΣΔ modulator outputs: SPI interface and Manchester coded 1-wire interface (both with adjustable parameters). DFSDM module supports the connection of up to 4 multiplexed input digital serial channels which are shared with up to 4 DFSDM modules. DFSDM module also supports alternative parallel data inputs from up to 4 internal 16-bit data channels (from internal ADCs or from device memory).

DFSDM is converting an input data stream into a final digital data word which represents an analog input value on a ΣΔ modulator analog input. The conversion is based on a configurable digital process: the digital filtering and decimation of the input serial data stream.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter type, filter order, length of filter, integrator length. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion mode and continuous mode. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A flexible timer triggering system can be used to control the start of conversion of DFSDM. This timing control is capable of triggering simultaneous conversions or inserting a programmable delay between conversions.

DFSDM features an analog watchdog function. Analog watchdog can be assigned to any of the input channel data stream or to final output data. Analog watchdog has its own digital filtering of input data stream to reach the required speed and resolution of watched data.

To detect short-circuit in control applications, there is a short-circuit detector. This block watches each input channel data stream for occurrence of stable data for a defined time duration (several 0’s or 1’s in an input data stream).

An extremes detector block watches final output data and stores maximum and minimum values from the output data values. The extremes values stored can be restarted by software.

Two power modes are supported: normal mode and stop mode.

26.2 DFSDM main features

- Up to 4 multiplexed input digital serial channels:
  - configurable SPI interface to connect various ΣΔ modulators
  - configurable Manchester coded 1 wire interface support
  - clock output for ΣΔ modulator(s)
- Alternative inputs from up to 4 internal digital parallel channels:
  - inputs with up to 16 bit resolution
  - internal sources: ADCs data or memory (CPU/DMA write) data streams
- Adjustable digital signal processing:
  - $Sinc^x$ filter: filter order/type (1..5), oversampling ratio (up to 1..1024)
  - integrator: oversampling ratio (1..256)
- Up to 24-bit output data resolution:
  - right bit-shifter on final data (0..31 bits)
- Signed output data format
- Automatic data offset correction (offset stored in register by user)
- Continuous or single conversion
- Start-of-conversion synchronization with:
  - software trigger
  - internal timers
  - external events
  - start-of-conversion synchronously with first DFSDM filter (DFSDM_FLT0)
- Analog watchdog feature:
  - low value and high value data threshold registers
  - own configurable $Sinc^x$ digital filter (order = 1..3, oversampling ratio = 1..32)
  - input from output data register or from one or more input digital serial channels
  - continuous monitoring independently from standard conversion
- Short-circuit detector to detect saturated analog input values (bottom and top ranges):
  - up to 8-bit counter to detect 1..256 consecutive 0's or 1's on input data stream
  - monitoring continuously each channel (4 serial channel transceiver outputs)
- Break generation on analog watchdog event or short-circuit detector event
- Extremes detector:
  - store minimum and maximum values of output data values
  - refreshed by software
- Pulse skipper feature to support beamforming applications (delay line like behavior)
- DMA may be used to read the conversion data
- Interrupts: end of conversion, overrun, analog watchdog, short-circuit, channel clock absence
- "regular" or "injected" conversions:
  - "regular" conversions can be requested at any time or even in continuous mode without having any impact on the timing of "injected" conversions
  - "injected" conversions for precise timing and with high conversion priority

26.3 DFSDM implementation

This section describes the configuration implemented in DFSDMx.

Table 197. DFSDM1 implementation

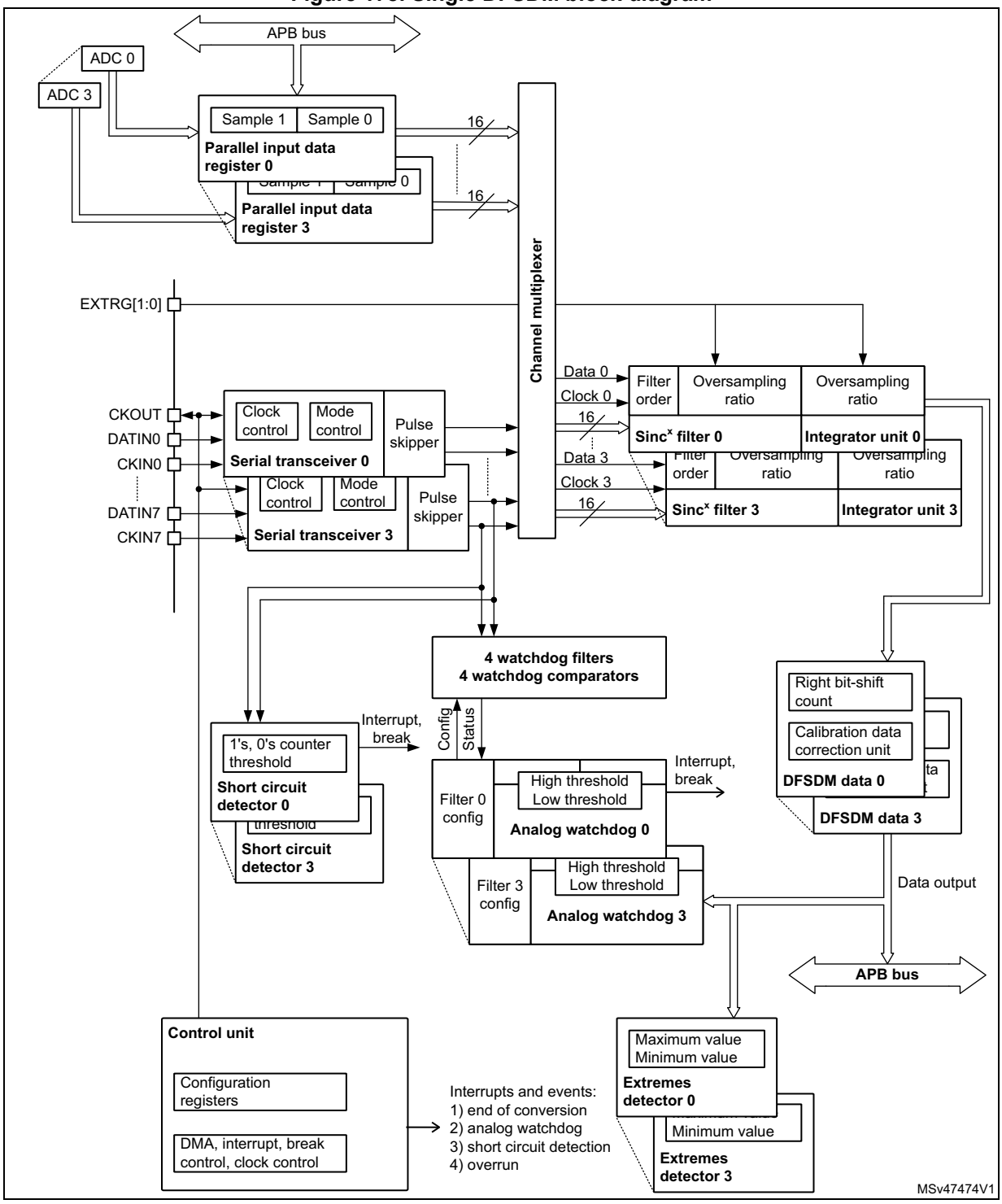
DFSDM features	DFSDM1
Number of channels	4
Number of filters	4
Input from internal ADC	X
Supported trigger sources	32(1)
Pulses skipper	X
ID registers support	-

1. Refer to *Table 200: DFSDM triggers connection* for available trigger sources.

26.4 DFSDM functional description

26.4.1 DFSDM block diagram

Figure 178. Single DFSDM block diagram



1. This example shows 4 DFSDM filters and 4 input channels.

26.4.2 DFSDM pins and internal signals

Table 198. DFSDM external pins

Name	Signal Type	Remarks
VDD	Power supply	Digital power supply.
VSS	Power supply	Digital ground power supply.
CKIN[3:0]	Clock input	Clock signal provided from external ΣΔ modulator. FT input.
DATIN[3:0]	Data input	Data signal provided from external ΣΔ modulator. FT input.
CKOUT	Clock output	Clock output to provide clock signal into external ΣΔ modulator.
EXTRG[1:0]	External trigger signal	Input trigger from two EXTI signals to start analog conversion (from GPIOs: EXTI11, EXTI15).

Table 199. DFSDM internal signals

Name	Signal Type	Remarks
dfsdm_jtrg[31:0]	Internal/ external trigger signal	Input trigger from internal/external trigger sources in order to start analog conversion (from internal sources: synchronous input, from external sources: asynchronous input with synchronization). See *Table 200* for details.
dfsdm_break[3:0]	break signal output	Break signals event generation from Analog watchdog or short-circuit detector
dfsdm_dma[3:0]	DMA request signal	DMA request signal from each DFSDM_FLTx (x=0..3): end of injected conversion event.
dfsdm_it[3:0]	Interrupt request signal	Interrupt signal for each DFSDM_FLTx (x=0..3)
dfsdm_dat_adc[15:0]	ADC input data	Up to 4 internal ADC data buses as parallel inputs.

Table 200. DFSDM triggers connection

Trigger name	Trigger source
dfsdm_jtrg0	TIM1_TRGO
dfsdm_jtrg1	TIM1_TRGO2
dfsdm_jtrg2	TIM8_TRGO
dfsdm_jtrg3	TIM8_TRGO2
dfsdm_jtrg[23:9]	Reserved
dfsdm_jtrg24	EXTI11
dfsdm_jtrg25	EXTI15
dfsdm_jtrg26	LPTIMER1
dfsdm_jtrg[31:27]	Reserved

Table 201. DFSDM break connection

Break name	Break destination
dfsdm_break[0]	TIM1/TIM15 break
dfsdm_break[1]	TIM1 break2 / TIM16 break
dfsdm_break[2]	TIM8/TIM17 break
dfsdm_break[3]	TIM8 break2

26.4.3 DFSDM reset and clocks

DFSDM on-off control

The DFSDM interface is globally enabled by setting DFSDMEN=1 in the DFSDM_CH0CFGR1 register. Once DFSDM is globally enabled, all input channels (y=0..3) and digital filters DFSDM_FLTx (x=0..3) start to work if their enable bits are set (channel enable bit CHEN in DFSDM_CHyCFGR1 and DFSDM_FLTx enable bit DFEN in DFSDM_FLTxCR1).

Digital filter x DFSDM_FLTx (x=0..3) is enabled by setting DFEN=1 in the DFSDM_FLTxCR1 register. Once DFSDM_FLTx is enabled (DFEN=1), both $Sinc^x$ digital filter unit and integrator unit are reinitialized.

By clearing DFEN, any conversion which may be in progress is immediately stopped and DFSDM_FLTx is put into stop mode. All register settings remain unchanged except DFSDM_FLTxAWSR and DFSDM_FLTxISR (which are reset).

Channel y (y=0..3) is enabled by setting CHEN=1 in the DFSDM_CHyCFGR1 register. Once the channel is enabled, it receives serial data from the external ΣΔ modulator or parallel internal data sources (ADCs or CPU/DMA wire from memory).

DFSDM must be globally disabled (by DFSDMEN=0 in DFSDM_CH0CFGR1) before stopping the system clock to enter in the STOP mode of the device.

DFSDM clocks

The internal DFSDM clock $f_{DFSDMCLK}$, which is used to drive the channel transceivers, digital processing blocks (digital filter, integrator) and next additional blocks (analog watchdog, short-circuit detector, extremes detector, control block) is generated by the RCC block and is derived from the system clock SYSCLK or peripheral clock PCLK2 (see *Section 9.8.32: RCC peripherals independent clock configuration register 2 (RCC_CCIPR2)*). The DFSDM clock is automatically stopped in stop mode (if DFEN = 0 for all DFSDM_FLTx, x=0..3).

The DFSDM serial channel transceivers can receive an external serial clock to sample an external serial data stream. The internal DFSDM clock must be at least 4 times faster than the external serial clock if standard SPI coding is used, and 6 times faster than the external serial clock if Manchester coding is used.

DFSDM can provide one external output clock signal to drive external ΣΔ modulator(s) clock input(s). It is provided on CKOUT pin. This output clock signal must be in the range specified in given device datasheet and is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM_CH0CFGR1 register) by programmable divider in the range 2 - 256 (CKOUTDIV in DFSDM_CH0CFGR1 register). Audio clock source is SAI1 clock selected by SAI1SEL[1:0] field in RCC configuration (see).

26.4.4 Serial channel transceivers

There are 4 multiplexed serial data channels which can be selected for conversion by each filter or Analog watchdog or Short-circuit detector. Those serial transceivers receive data stream from external ΣΔ modulator. Data stream can be sent in SPI format or Manchester coded format (see SITP[1:0] bits in DFSDM_CHyCFGR1 register).
The channel is enabled for operation by setting CHEN=1 in DFSDM_CHyCFGR1 register.

Channel inputs selection

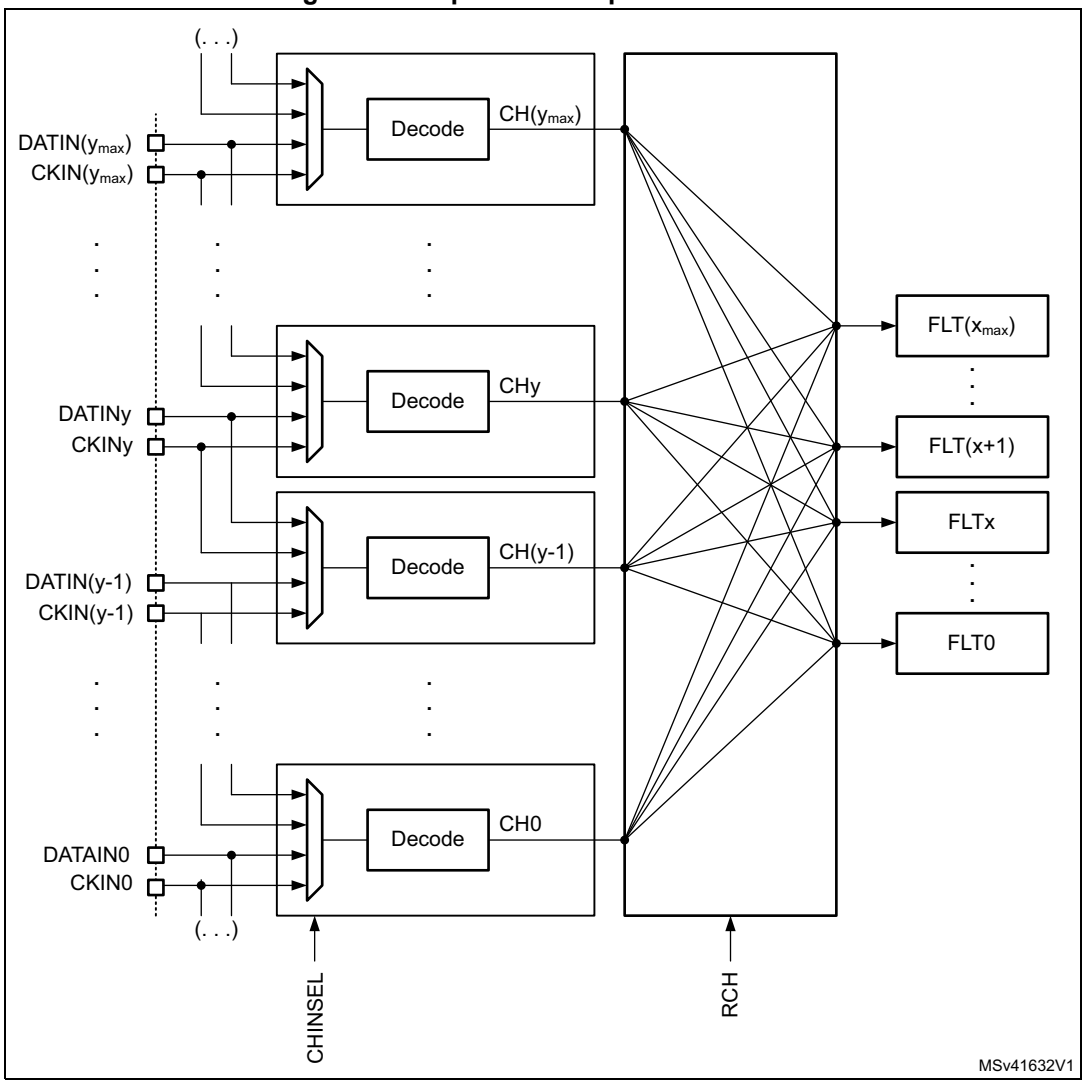
Serial inputs (data and clock signals) from DATINy and CKINy pins can be redirected from the following channel pins. This serial input channel redirection is set by CHINSEL bit in DFSDM_CHyCFGR1 register.

Channel redirection can be used to collect audio data from PDM (pulse density modulation) stereo microphone type. PDM stereo microphone has one data and one clock signal. Data signal provides information for both left and right audio channel (rising clock edge samples for left channel and falling clock edge samples for right channel).

Configuration of serial channels for PDM microphone input:

- PDM microphone signals (data, clock) will be connected to DFSDM input serial channel y (DATINy, CKOUT) pins.
- Channel y will be configured: CHINSEL = 0 (input from given channel pins: DATINy, CKINy).
- Channel (y-1) (modulo 4) will be configured: CHINSEL = 1 (input from the following channel ((y-1)+1) pins: DATINy, CKINy).
- Channel y: SITP[1:0] = 0 (rising edge to strobe data) => left audio channel on channel y.
- Channel (y-1): SITP[1:0] = 1 (falling edge to strobe data) => right audio channel on channel y-1.
- Two DFSDM filters will be assigned to channel y and channel (y-1) (to filter left and right channels from PDM microphone).

Figure 179. Input channel pins redirection



Output clock generation

A clock signal can be provided on CKOUT pin to drive external ΣΔ modulator clock inputs. The frequency of this CKOUT signal is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM_CH0CFGR1 register) divided by a predivider (see CKOUTDIV bits in DFSDM_CH0CFGR1 register). If the output clock is stopped, then CKOUT signal is set to low state (output clock can be stopped by CKOUTDIV=0 in DFSDM_CHyCFGR1 register or by DFSDMEN=0 in DFSDM_CH0CFGR1 register). The output clock stopping is performed:

- 4 system clocks after DFSDMEN is cleared (if CKOUTSRC=0)
- 1 system clock and 3 audio clocks after DFSDMEN is cleared (if CKOUTSRC=1)

Before changing CKOUTSRC the software has to wait for CKOUT being stopped to avoid glitch on CKOUT pin. The output clock signal frequency must be in the range 0 - 20 MHz.

SPI data input format operation

In SPI format, the data stream is sent in serial format through data and clock signals. Data signal is always provided from DATINy pin. A clock signal can be provided externally from CKINy pin or internally from a signal derived from the CKOUT signal source.

In case of external clock source selection (SPICKSEL[1:0]=0) data signal (on DATINy pin) is sampled on rising or falling clock edge (of CKINy pin) according SITP[1:0] bits setting (in DFSDM_CHyCFGR1 register).

Internal clock sources - see SPICKSEL[1:0] in DFSDM_CHyCFGR1 register:

- CKOUT signal:
  - For connection to external ΣΔ modulator which uses directly its clock input (from CKOUT) to generate its output serial communication clock.
  - Sampling point: on rising/falling edge according SITP[1:0] setting.
- CKOUT/2 signal (generated on CKOUT rising edge):
  - For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
  - Sampling point: on each second CKOUT falling edge.
- CKOUT/2 signal (generated on CKOUT falling edge):
  - For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).
  - Sampling point: on each second CKOUT rising edge.

Note: An internal clock source can only be used when the external ΣΔ modulator uses CKOUT signal as a clock input (to have synchronous clock and data operation).

Internal clock source usage can save CKINy pin connection (CKINy pins can be used for other purpose).

The clock source signal frequency must be in the range 0 - 20 MHz for SPI coding and less than $f_{DFSDMCLK}/4$.

Manchester coded data input format operation

In Manchester coded format, the data stream is sent in serial format through DATINy pin only. Decoded data and clock signal are recovered from serial stream after Manchester decoding. There are two possible settings of Manchester codings (see SITP[1:0] bits in DFSDM_CHyCFGR1 register):

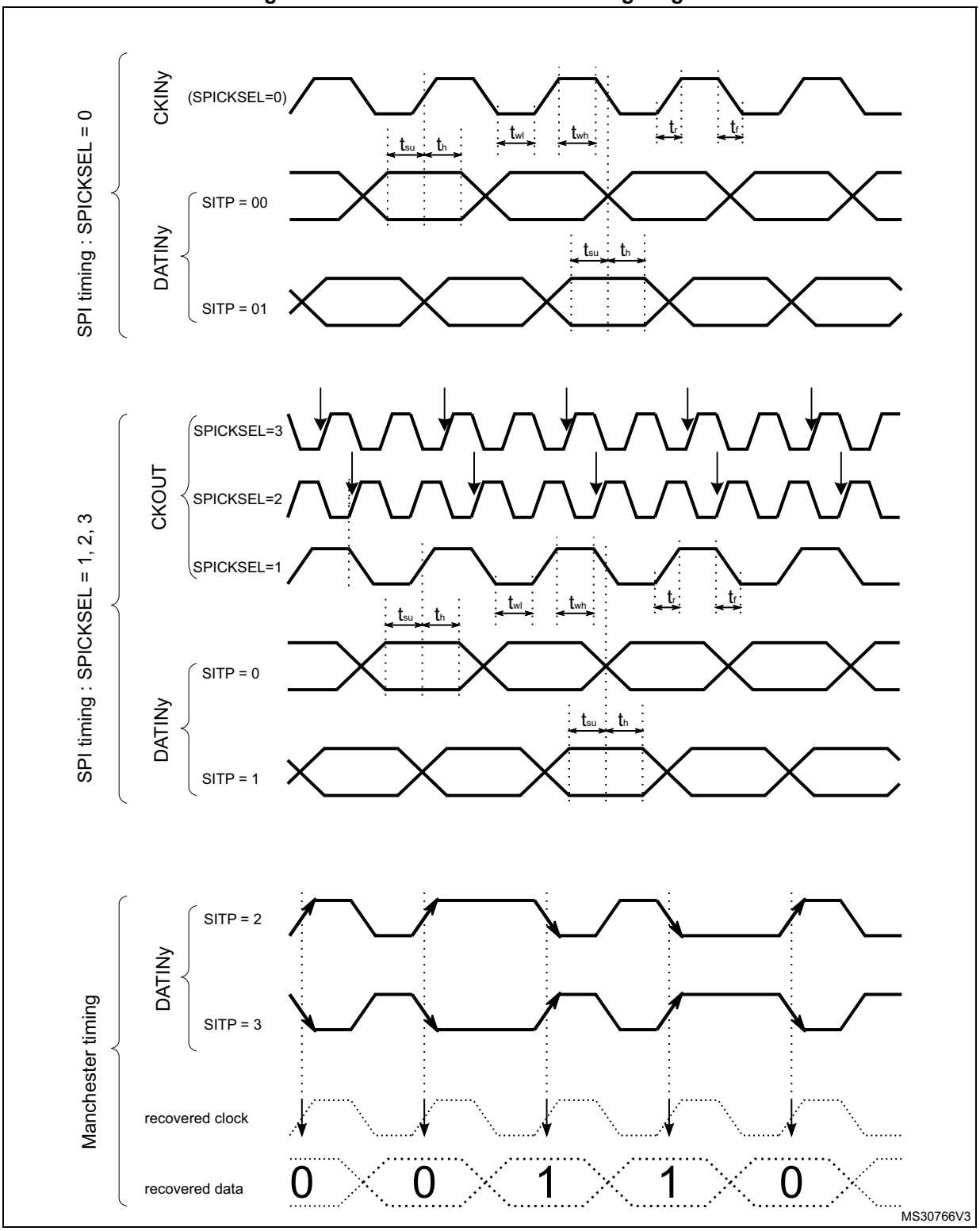
- signal rising edge = log 0; signal falling edge = log 1
- signal rising edge = log 1; signal falling edge = log 0

The recovered clock signal frequency for Manchester coding must be in the range 0 - 10 MHz and less than $f_{DFSDMCLK}/6$.

To correctly receive Manchester coded data, the CKOUTDIV divider (in DFSDM_CH0CFGR1 register) must be set with respect to expected Manchester data rate according formula:

$$((CKOUTDIV + 1) \times T_{SYSCLK}) < T_{Manchester\ clock} < (2 \times CKOUTDIV \times T_{SYSCLK})$$

Figure 180. Channel transceiver timing diagrams



Clock absence detection

Channels serial clock inputs can be checked for clock absence/presence to ensure the correct operation of conversion and error reporting. Clock absence detection can be enabled or disabled on each input channel y by bit CKABEN in DFSDM_CHyCFGR1 register. If enabled, then this clock absence detection is performed continuously on a given channel. A clock absence flag is set (CKABF[y] = 1) and an interrupt can be invoked (if CKABIE=1) in case of an input clock error (see CKABF[3:0] in DFSDM_FLT0ISR register and CKABEN in DFSDM_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM_FLT0ICR register), the clock absence flag is refreshed. Clock absence status bit CKABF[y] is set also by hardware when corresponding channel y is disabled (if CHEN[y] = 0 then CKABF[y] is held in set state).

When a clock absence event has occurred, the data conversion (and/or analog watchdog and short-circuit detector) provides incorrect data. The user should manage this event and discard given data while a clock absence is reported.

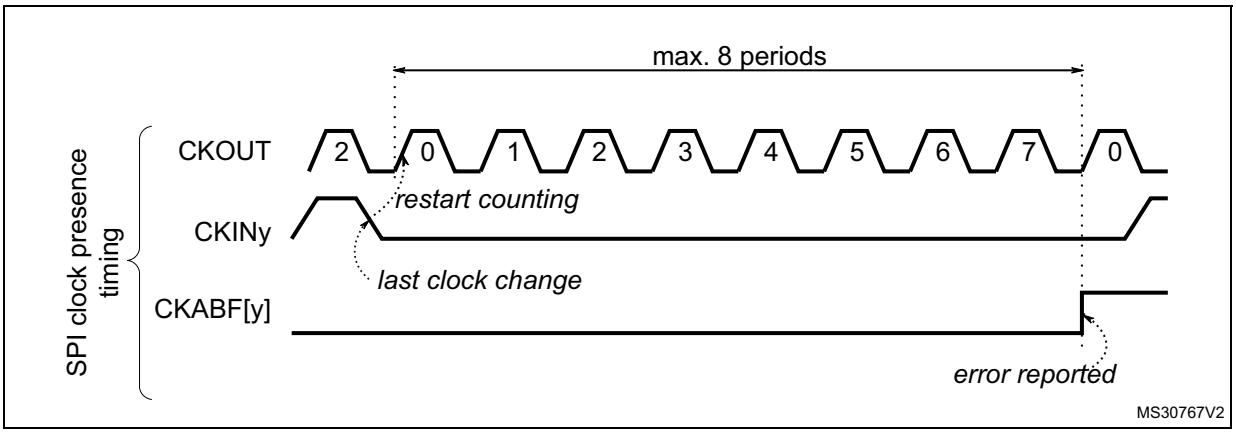
The clock absence feature is available only when the system clock is used for the CKOUT signal (CKOUTSRC=0 in DFSDM_CH0CFGR1 register).

When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y] bit (in DFSDM_FLT0ICR register). The software sequence concerning clock absence detection feature should be:

- Enable given channel by CHEN = 1
- Try to clear the clock absence flag (by CLRCKABF = 1) until the clock absence flag is really cleared (CKABF = 0). At this time, the transceiver is synchronized (signal clock is valid) and is able to receive data.
- Enable the clock absence feature CKABEN = 1 and the associated interrupt CKABIE = 1 to detect if the SPI clock is lost or Manchester data edges are missing.

If SPI data format is used, then the clock absence detection is based on the comparison of an external input clock with an output clock generation (CKOUT signal). The external input clock signal into the input channel must be changed at least once per 8 signal periods of CKOUT signal (which is controlled by CKOUTDIV field in DFSDM_CH0CFGR1 register).

Figure 181. Clock absence timing diagram for SPI



If Manchester data format is used, then the clock absence means that the clock recovery is unable to perform from Manchester coded signal. For a correct clock recovery, it is first necessary to receive data with 1 to 0 or 0 to 1 transition (see *Figure 183* for Manchester synchronization).

The detection of a clock absence in Manchester coding (after a first successful synchronization) is based on changes comparison of coded serial data input signal with output clock generation (CKOUT signal). There must be a voltage level change on DATINy pin during 2 periods of CKOUT signal (which is controlled by CKOUTDIV bits in DFSDM_CH0CFGR1 register). This condition also defines the minimum data rate to be able to correctly recover the Manchester coded data and clock signals.

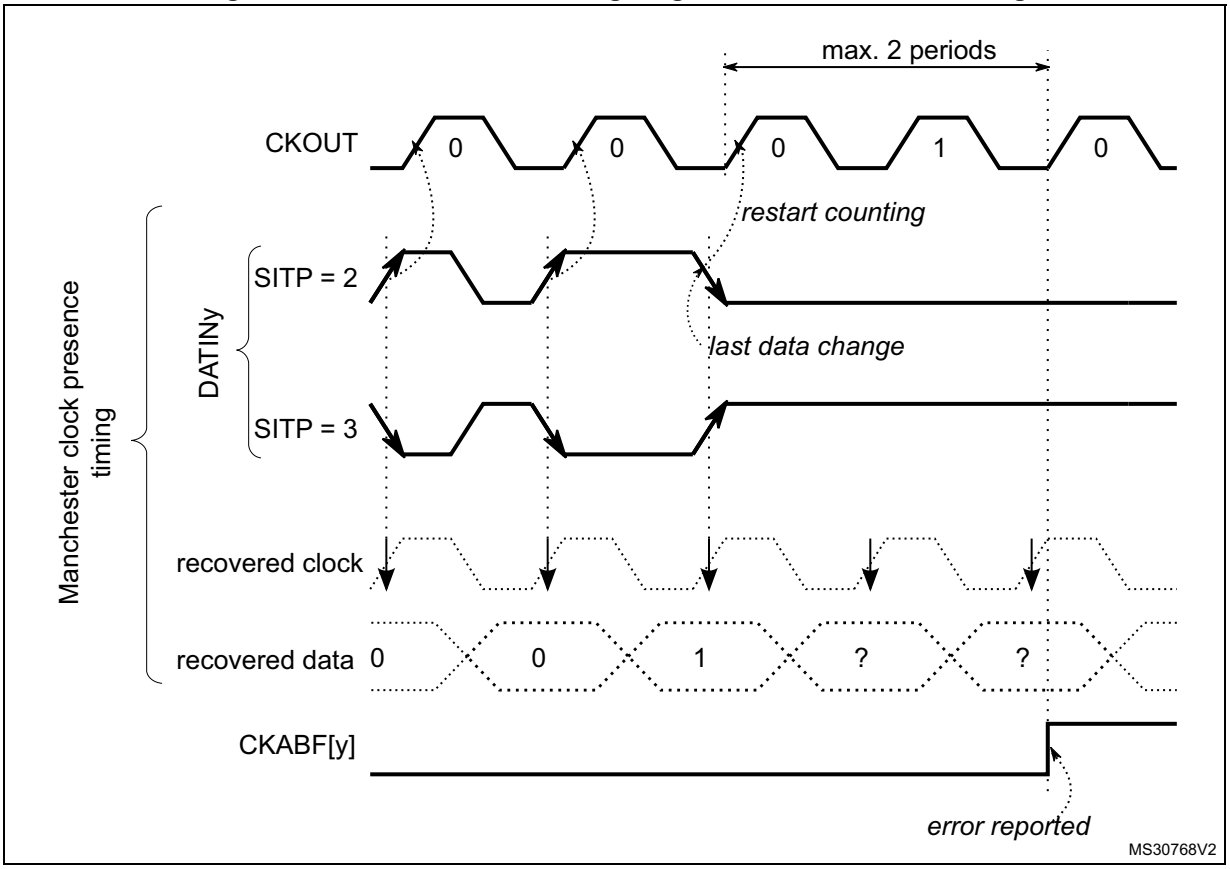
The maximum data rate of Manchester coded data must be less than the CKOUT signal.

So to correctly receive Manchester coded data, the CKOUTDIV divider must be set according the formula:

$$((CKOUTDIV + 1) \times T_{SYSCLK}) < T_{Manchester\ clock} < (2 \times CKOUTDIV \times T_{SYSCLK})$$

A clock absence flag is set (CKABF[y] = 1) and an interrupt can be invoked (if CKABIE=1) in case of an input clock recovery error (see CKABF[3:0] in DFSDM_FLT0ISR register and CKABEN in DFSDM_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM_FLT0ICR register), the clock absence flag is refreshed.

Figure 182. Clock absence timing diagram for Manchester coding



Manchester/SPI code synchronization

The Manchester coded stream must be synchronized the first time after enabling the channel (CHEN=1 in DFSDM_CHyCFGR1 register). The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (to be able to detect valid data edge) is received. The end of the synchronization can be checked by polling CKABF[y]=0 for a given channel after it has been cleared by CLRCKABF[y] in DFSDM_FLT0ICR, following the software sequence detailed hereafter:

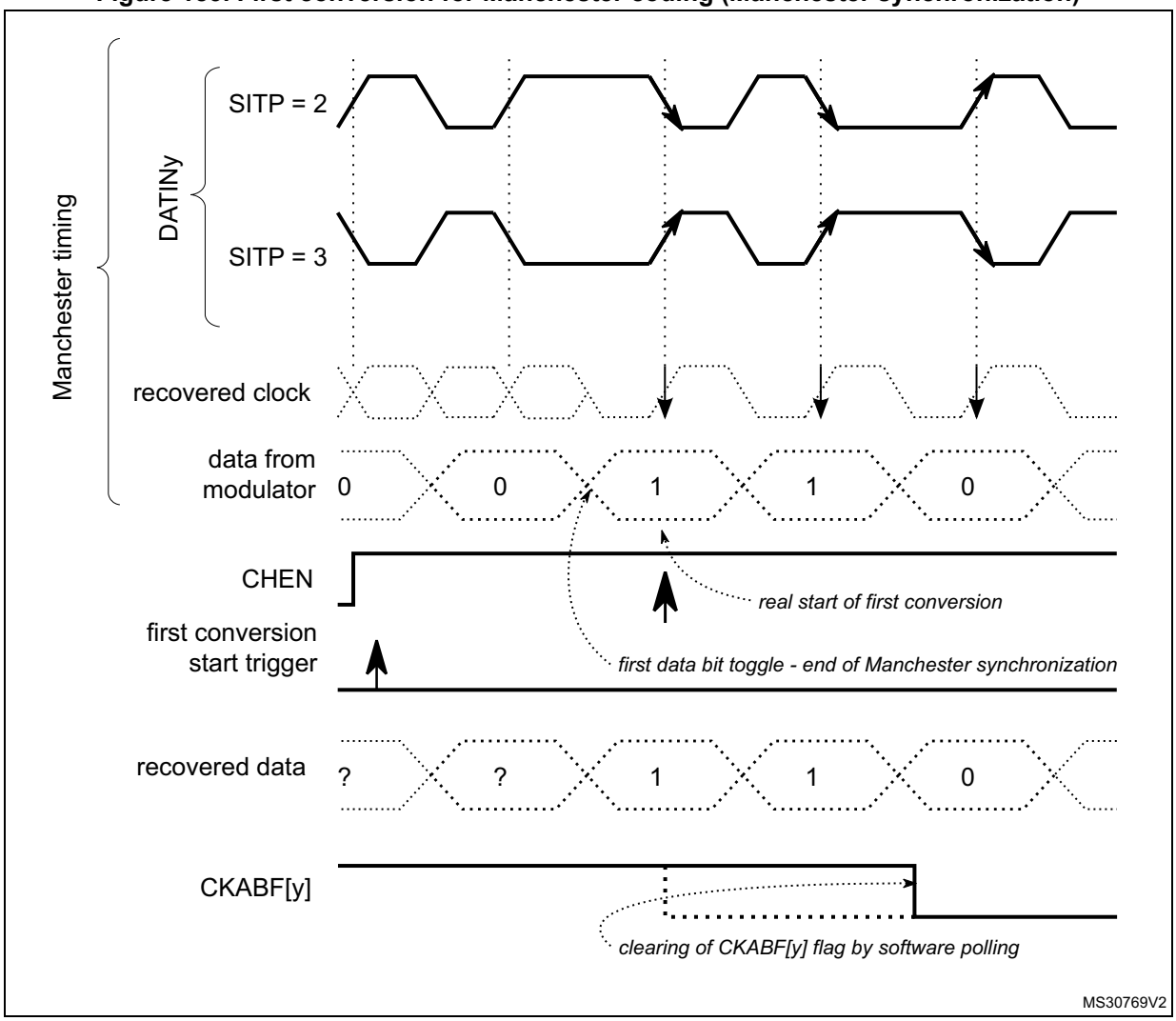
CKABF[y] flag is cleared by setting CLRCKABF[y] bit. If channel y is not yet synchronized the hardware immediately set the CKABF[y] flag. Software is then reading back the CKABF[y] flag and if it is set then perform again clearing of this flag by setting CLRCKABF[y] bit. This software sequence (polling of CKABF[y] flag) continues until CKABF[y] flag is set (signalizing that Manchester stream is synchronized). To be able to synchronize/receive Manchester coded data the CKOUTDIV divider (in DFSDM_CH0CFGR1 register) must be set with respect to expected Manchester data rate according the formula below.

$$((CKOUTDIV + 1) \times T_{SYSCLK}) < T_{Manchester\ clock} < (2 \times CKOUTDIV \times T_{SYSCLK})$$

SPI coded stream is synchronized after first detection of clock input signal (valid rising/falling edge).

Note: When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y] bit (in DFSDM_FLT0ICR register).

Figure 183. First conversion for Manchester coding (Manchester synchronization)



External serial clock frequency measurement

The measuring of a channel serial clock input frequency provides a real data rate from an external ΣΔ modulator, which is important for application purposes.

An external serial clock input frequency can be measured by a timer counting DFSDM clocks ($f_{DFSDMCLK}$) during one conversion duration. The counting starts at the first input data clock after a conversion trigger (regular or injected) and finishes by last input data clock before conversion ends (end of conversion flag is set). Each conversion duration (time between first serial sample and last serial sample) is updated in counter CNVCNT[27:0] in register DFSDM_FLTxCNVTIMR when the conversion finishes (JEOCF=1 or REOCF=1). The user can then compute the data rate according to the digital filter settings (FORD, FOSR, IOSR, FAST). The external serial frequency measurement is stopped only if the filter is bypassed (FOSR=0, only integrator is active, CNVCNT[27:0]=0 in DFSDM_FLTxCNVTIMR register).

In case of parallel data input (*Section 26.4.6: Parallel data inputs*) the measured frequency is the average input data rate during one conversion.

Note: When conversion is interrupted (e.g. by disabling/enabling the selected channel) the interruption time is also counted in CNVCNT[27:0]. Therefore it is recommended to not interrupt the conversion for correct conversion duration result.

Conversion times:

injected conversion or regular conversion with FAST = 0 (or first conversion if FAST=1):

for $Sinc^x$ filters (x=1..5):

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$

for FastSinc filter:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + 4) + 2] / f_{CKIN}$

regular conversion with FAST = 1 (except first conversion):

for $Sinc^x$ and FastSinc filters:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case if F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator):

$t = I_{OSR} / f_{CKIN}$ (... but CNVCNT=0)

where:

- f_{CKIN} is the channel input clock frequency (on given channel CKINy pin) or input data rate (in case of parallel data input)
- F_{OSR} is the filter oversampling ratio: F_{OSR} = FOSR[9:0]+1 (see DFSDM_FLTxFCR register)
- I_{OSR} is the integrator oversampling ratio: I_{OSR} = IOSR[7:0]+1 (see DFSDM_FLTxFCR register)
- F_{ORD} is the filter order: F_{ORD} = FORD[2:0] (see DFSDM_FLTxFCR register)

Channel offset setting

Each channel has its own offset setting (in register) which is finally subtracted from each conversion result (injected or regular) from a given channel. Offset correction is performed after the data right bit shift. The offset is stored as a 24-bit signed value in OFFSET[23:0] field in DFSDM_CHyCFGR2 register.

Data right bit shift

To have the result aligned to a 24-bit value, each channel defines a number of right bit shifts which will be applied on each conversion result (injected or regular) from a given channel. The data bit shift number is stored in DTRBS[4:0] bits in DFSDM_CHyCFGR2 register.

The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained, in order to have valid 24-bit signed format of result data.

Pulses skipper

Purpose of the pulses skipper is to implement delay line like behavior for given input channel(s). Given number of samples from input serial data stream (serial stream only) can be discarded before they enter into the filter. This data discarding is performed by skipping given number of sampling input clock pulses (given serial data samples are then not sampled by filter). The sampling clock is gated by pulses skipper function for given number of clock pulses. When given clock pulses are skipped then the filtering continues for following input data. With comparison to non skipped data stream this operation causes that

the final output sample (and next samples) from filter will be calculated from later input data. This final sample then looks a bit in forward - because it is calculated from newer input samples than the “non-skipped” sample. The final “skipped sample” is converted later because the skipped input data samples must be replaced by followed input data samples. The final data buffers behavior (skipped and non-skipped output data buffers comparison) looks like the non-skipped data stream is a bit delayed - both data buffers will be phase shifted.

Number of clock pulses to be skipped should be written into PLSSKP[5:0] field in DFSDM_CHyDLYR register. Once PLSSKP[5:0] field is written the execution of pulses skipping is started on given channel. PLSSKP[5:0] field can be read in order to check the progress of pulses skipper. When PLSSKP[5:0]=0 means that pulses skipping has been executed.

Up to 63 clock pulses can be skip with a single write operation into PLSSKP[5:0]. If more pulses need to be skipped, then user has to write several times into the PLSSKP[5:0] field. The application software should handle cumulative skipped clock number per each filter.

26.4.5 Configuring the input serial interface

The following parameters must be configured for the input serial interface:

- **Output clock predivider**. There is a programmable predivider to generate the output clock from DFSDM clock (2 - 256). It is defined by CKOUTDIV[7:0] bits in DFSDM_CH0CFGR1 register.
- **Serial interface type and input clock phase.** Selection of SPI or Manchester coding and sampling edge of input clock. It is defined by SITP [1:0] bits in DFSDM_CHyCFGR1 register.
- **Input clock source.** External source from CKINy pin or internal from CKOUT pin. It is defined by SPICKSEL[1:0] field in DFSDM_CHyCFGR1 register.
- **Final data right bit-shift.** Defines the final data right bit shift to have the result aligned to a 24-bit value. It is defined by DTRBS[4:0] in DFSDM_CHyCFGR2 register.
- **Channel offset per channel.** Defines the analog offset of a given serial channel (offset of connected external ΣΔ modulator). It is defined by OFFSET[23:0] bits in DFSDM_CHyCFGR2 register.
- **short-circuit detector and clock absence per channel enable.** To enable or disable the short-circuit detector (by SCDEN bit) and the clock absence monitoring (by CKABEN bit) on a given serial channel in register DFSDM_CHyCFGR1.
- **Analog watchdog filter and short-circuit detector threshold settings.** To configure channel analog watchdog filter parameters and channel short-circuit detector parameters. Configurations are defined in DFSDM_CHyAWSCDR register.

26.4.6 Parallel data inputs

Each input channel provides a register for 16-bit parallel data input (besides serial data input). Each 16-bit parallel input can be sourced from internal data sources only:

- internal ADC results
- direct CPU/DMA writing.

The selection for using serial or parallel data input for a given channel is done by field DATMPX[1:0] of DFSDM_CHyCFGR1 register. In DATMPX[1:0] is also defined the parallel data source: internal ADC or direct write by CPU/DMA.

Each channel contains a 32-bit data input register DFSDM_CHyDATINR in which it can be written a 16-bit data. Data are in 16-bit signed format. Those data can be used as input to the digital filter which is accepting 16-bit parallel data.

If serial data input is selected (DATMPX[1:0] = 0), the DFSDM_CHyDATINR register is write protected.

Input from internal ADC

In case of ADC data parallel input (DATMPX[1:0]=1) the ADC[y+1] result is assigned to channel y input (ADC1 is filling DFSDM_CHDATIN0R register, ADC2 is filling DFSDM_CHDATIN1R register, ... , ADC8 is filling DFSDM_CHDATIN7R register). End of conversion event from ADC[y+1] causes update of channel y data (parallel data from ADC[y+1] are put as next sample to digital filter). Data from ADC[y+1] is written into DFSDM_CHyDATINR register (field INDAT0[15:0]) when end of conversion event occurred.

The setting of data packing mode (DATPACK[1:0] in the DFSDM_CHyCFGR1 register) has no effect in case of ADC data input.

Note: Extension of ADC specification: in case the internal ADC is configured in interleaved mode (e.g. ADC1 together with ADC2 - see ADC specification) then each result from ADC1 or from ADC2 will come to the same 16-bit bus - to the bus of ADC1 - which is coming into DFSDM channel 0 (fixed connection). So there will be double input data rate into DFSDM channel 0 (even samples come from ADC1 and odd samples from ADC2). Channel 1 associated with ADC2 will be free.

Input from memory (direct CPU/DMA write)

The direct data write into DFSDM_CHyDATINR register by CPU or DMA (DATMPX[1:0]=2) can be used as data input in order to process digital data streams from memory or peripherals.

Data can be written by CPU or DMA into DFSDM_CHyDATINR register:

1. **CPU data write:**

   Input data are written directly by CPU into DFSDM_CHyDATINR register.

2. **DMA data write:**

   The DMA should be configured in memory-to-memory transfer mode to transfer data from memory buffer into DFSDM_CHyDATINR register. The destination memory address is the address of DFSDM_CHyDATINR register. Data are transferred at DMA transfer speed from memory to DFSDM parallel input.

   This DMA transfer is different from DMA used to read DFSDM conversion results. Both DMA can be used at the same time - first DMA (configured as memory-to-memory transfer) for input data writings and second DMA (configured as peripheral-to-memory transfer) for data results reading.

The accesses to DFSDM_CHyDATINR can be either 16-bit or 32-bit wide, allowing to load respectively one or two samples in one write operation. 32-bit input data register (DFSDM_CHyDATINR) can be filled with one or two 16-bit data samples, depending on the data packing operation mode defined in field DATPACK[1:0] of DFSDM_CHyCFGR1 register:

1. **Standard mode (DATPACK[1:0]=0):**

   Only one sample is stored in field INDAT0[15:0] of DFSDM_CHyDATINR register which is used as input data for channel y. The upper 16 bits (INDAT1[15:0]) are ignored and write protected. The digital filter must perform one input sampling (from INDAT0[15:0])

to empty data register after it has been filled by CPU/DMA. This mode is used together with 16-bit CPU/DMA access to DFSDM_CHyDATINR register to load one sample per write operation.

2. **Interleaved mode (DATPACK[1:0]=1):**

   DFSDM_CHyDATINR register is used as a two sample buffer. The first sample is stored in INDAT0[15:0] and the second sample is stored in INDAT1[15:0]. The digital filter must perform two input samplings from channel y to empty DFSDM_CHyDATINR register. This mode is used together with 32-bit CPU/DMA access to DFSDM_CHyDATINR register to load two samples per write operation.

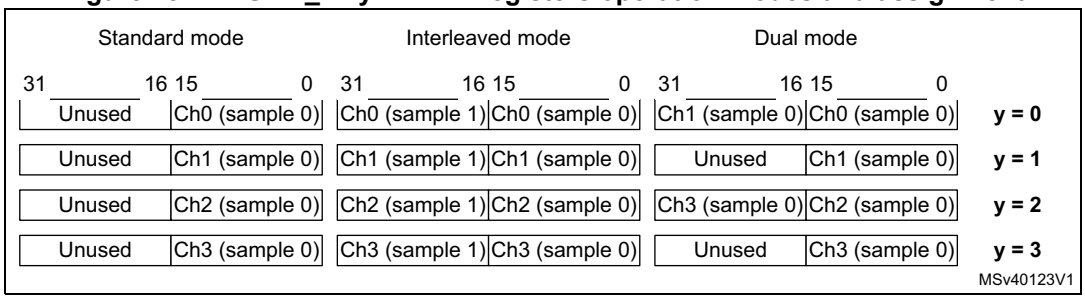
3. **Dual mode (DATPACK[1:0]=2):**

   Two samples are written into DFSDM_CHyDATINR register. The data INDAT0[15:0] is for channel y, the data in INDAT1[15:0] is for channel y+1. The data in INDAT1[15:0] is automatically copied INDAT0[15:0] of the following (y+1) channel data register DFSDM_CH[y+1]DATINR). The digital filters must perform two samplings - one from channel y and one from channel (y+1) - in order to empty DFSDM_CHyDATINR registers.

   Dual mode setting (DATPACK[1:0]=2) is available only on even channel numbers (y = 0, 2). If odd channel (y = 1, 3) is set to Dual mode then both INDAT0[15:0] and INDAT1[15:0] parts are write protected for this channel. If even channel is set to Dual mode then the following odd channel must be set into Standard mode (DATPACK[1:0]=0) for correct cooperation with even channels.

See *Figure 184* for DFSDM_CHyDATINR registers data modes and assignments of data samples to channels.

Figure 184. DFSDM_CHyDATINR registers operation modes and assignment



The write into DFSDM_CHyDATINR register to load one or two samples must be performed after the selected input channel (channel y) is enabled for data collection (starting conversion for channel y). Otherwise written data are lost for next processing.

For example: for single conversion and interleaved mode, do not start writing pair of data samples into DFSDM_CHyDATINR before the single conversion is started (any data present in the DFSDM_CHyDATINR before starting a conversion is discarded).

26.4.7 Channel selection

There are 4 multiplexed channels which can be selected for conversion using the injected channel group and/or using the regular channel.

The **injected channel group** is a selection of any or all of the 4 channels. JCHG[3:0] in the DFSDM_FLTxJCHGR register selects the channels of the injected group, where JCHG[y]=1 means that channel y is selected.

Injected conversions can operate in scan mode (JSCAN=1) or single mode (JSCAN=0). In scan mode, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first, followed immediately by the next higher channel until all the channels selected by JCHG[3:0] have been converted. In single mode (JSCAN=0), only one channel from the selected channels is converted, and the channel selection is moved to the next channel. Writing to JCHG[3:0] if JSCAN=0 resets the channel selection to the lowest selected channel.

Injected conversions can be launched by software or by a trigger. They are never interrupted by regular conversions.

The **regular channel** is a selection of just one of the 4 channels. RCH[1:0] in the DFSDM_FLTxCR1 register indicates the selected channel.

Regular conversions can be launched only by software (not by a trigger). A sequence of continuous regular conversions is temporarily interrupted when an injected conversion is requested.

Performing a conversion on a disabled channel (CHEN=0 in DFSDM_CHyCFGR1 register) causes that the conversion will never end - because no input data is provided (with no clock signal). In this case, it is necessary to enable a given channel (CHEN=1 in DFSDM_CHyCFGR1 register) or to stop the conversion by DFEN=0 in DFSDM_FLTxCR1 register.

26.4.8 Digital filter configuration

DFSDM contains a $Sinc^x$ type digital filter implementation. This $Sinc^x$ filter performs an input digital data stream filtering, which results in decreasing the output data rate (decimation) and increasing the output data resolution. The $Sinc^x$ digital filter is configurable in order to reach the required output data rates and required output data resolution. The configurable parameters are:

- Filter order/type: (see FORD[2:0] bits in DFSDM_FLTxFCR register):
  - FastSinc
  - $Sinc^1$
  - $Sinc^2$
  - $Sinc^3$
  - $Sinc^4$
  - $Sinc^5$
- Filter oversampling/decimation ratio (see FOSR[9:0] bits in DFSDM_FLTxFCR register):
  - FOSR = 1-1024 - for FastSinc filter and $Sinc^x$ filter x = F_{ORD} = 1..3
  - FOSR = 1-215 - for $Sinc^x$ filter x = F_{ORD} = 4
  - FOSR = 1-73 - for $Sinc^x$ filter x = F_{ORD} = 5

The filter has the following transfer function (impulse response in H domain):

- $Sinc^x$ filter type: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^x$$
- FastSinc filter type: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

Figure 185. Example: Sinc^3 filter response

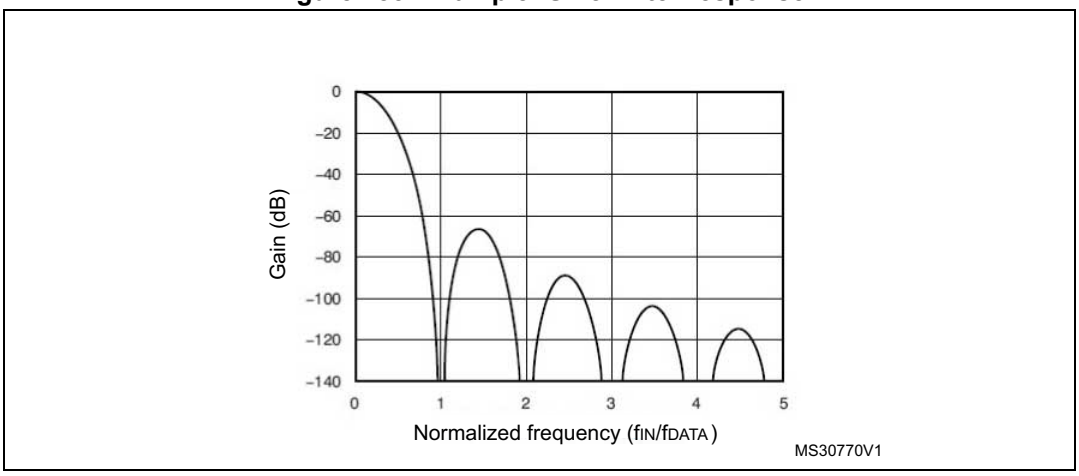


Table 202. Filter maximum output resolution (peak data values from filter output) for some FOSR values

FOSR	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5
x	+/- x	+/- x^2	+/- $2x^2$	+/- x^3	+/- x^4	+/- x^5
4	+/- 4	+/- 16	+/- 32	+/- 64	+/- 256	+/- 1024
8	+/- 8	+/- 64	+/- 128	+/- 512	+/- 4096	-
32	+/- 32	+/- 1024	+/- 2048	+/- 32768	+/- 1048576	+/- 33554432
64	+/- 64	+/- 4096	+/- 8192	+/- 262144	+/- 16777216	+/- 1073741824
128	+/- 128	+/- 16384	+/- 32768	+/- 2097152	+/- 268435456	Result can overflow on full scale input (> 32-bit signed integer)
256	+/- 256	+/- 65536	+/- 131072	+/- 16777216		
1024	+/- 1024	+/- 1048576	+/- 2097152	+/- 1073741824		

For more information about Sinc filter type properties and usage, it is recommended to study the theory about digital filters (more resources can be downloaded from internet).

26.4.9 Integrator unit

The integrator performs additional decimation and a resolution increase of data coming from the digital filter. The integrator simply performs the sum of data from a digital filter for a given number of data samples from a filter.

The integrator oversampling ratio parameter defines how many data counts will be summed to one data output from the integrator. IOSR can be set in the range 1-256 (see IOSR[7:0] bits description in DFSDM_FLTxFCR register).

Table 203. Integrator maximum output resolution (peak data values from integrator output) for some IOSR values and FOSR = 256 and Sinc^3 filter type (largest data)

IOSR	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5
x	+/- FOSR. x	+/- FOSR^2. x	+/- 2.FOSR^2. x	+/- FOSR^3. x	+/- FOSR^4. x	+/- FOSR^5. x
4	-	-	-	+/- 67 108 864	-	-
32	-	-	-	+/- 536 870 912	-	-
128	-	-	-	+/- 2 147 483 648	-	-
256	-	-	-	+/- 2^{32}	-	-

26.4.10 Analog watchdog

The analog watchdog purpose is to trigger an external signal (break or interrupt) when an analog signal reaches or crosses given maximum and minimum threshold values. An interrupt/event/break generation can then be invoked.

Each analog watchdog will supervise serial data receiver outputs (after the analog watchdog filter on each channel) or data output register (current injected or regular conversion result) according to AWFSEL bit setting (in DFSDM_FLTxCR1 register). The input channels to be monitored or not by the analog watchdog x will be selected by AWDCH[3:0] in DFSDM_FLTxCR2 register.

Analog watchdog conversions on input channels are independent from standard conversions. In this case, the analog watchdog uses its own filters and signal processing on each input channel independently from the main injected or regular conversions. Analog watchdog conversions are performed in a continuous mode on the selected input channels in order to watch channels also when main injected or regular conversions are paused (RCIP = 0, JCIP = 0).

There are high and low threshold registers which are compared with given data values (set by AWHT[23:0] bits in DFSDM_FLTxAWHTR register and by AWLT[23:0] bits in DFSDM_FLTxAWLTR register).

There are 2 options for comparing the threshold registers with the data values

- Option1: in this case, the input data are taken from final output data register (AWFSEL=0). This option is characterized by:
  - high input data resolution (up to 24-bits)
  - slow response time - inappropriate for fast response applications like overcurrent detection
  - for the comparison the final data are taken after bit shifting and offset data correction
  - final data are available only after main regular or injected conversions are performed
  - can be used in case of parallel input data source (DATMPX[1:0] ≠ 0 in DFSDM_CHyCFGR1 register)
- Option2: in this case, the input data are taken from any serial data receivers output (AWFSEL=1). This option is characterized by:
  - input serial data are processed by dedicated analog watchdog $Sinc^x$ channel filters with configurable oversampling ratio (1..32) and filter order (1..3) (see AWFOSR[4:0] and AWFORD[1:0] bits setting in DFSDM_CHyAWSCDR register)
  - lower resolution (up to 16-bit)
  - fast response time - appropriate for applications which require a fast response like overcurrent/overvoltage detection)
  - data are available in continuous mode independently from main regular or injected conversions activity

In case of input channels monitoring (AWFSEL=1), the data for comparison to threshold is taken from channels selected by AWDCH[3:0] field (DFSDM_FLTxCR2 register). Each of the selected channels filter result is compared to one threshold value pair (AWHT[23:0] / AWLT[23:0]). In this case, only higher 16 bits (AWHT[23:8] / AWLT[23:8]) define the 16-bit threshold compared with the analog watchdog filter output because data coming from the analog watchdog filter is up to a 16-bit resolution. Bits AWHT[7:0] / AWLT[7:0] are not taken into comparison in this case (AWFSEL=1).

Parameters of the analog watchdog filter configuration for each input channel are set in DFSDM_CHyAWSCDR register (filter order AWFORD[1:0] and filter oversampling ratio AWFOSR[4:0]).

Each input channel has its own comparator which compares the analog watchdog data (from analog watchdog filter) with analog watchdog threshold values (AWHT/AWLT). When several channels are selected (field AWDCH[3:0] field of DFSDM_FLTxCR2 register), several comparison requests may be received simultaneously. In this case, the channel request with the lowest number is managed first and then continuing to higher selected channels. For each channel, the result can be recorded in a separate flag (fields AWHTF[3:0], AWLTF[3:0] of DFSDM_FLTxAWSR register). Each channel request is executed in 8 DFSDM clock cycles. So, the bandwidth from each channel is limited to 8 DFSDM clock cycles (if AWDCH[3:0] = 0x0F). Because the maximum input channel sampling clock frequency is the DFSDM clock frequency divided by 4, the configuration AWFOSR = 0 (analog watchdog filter is bypassed) cannot be used for analog watchdog feature at this input clock speed. Therefore user must properly configure the number of watched channels and analog watchdog filter parameters with respect to input sampling clock speed and DFSDM frequency.

Analog watchdog filter data for given channel y is available for reading by firmware on field WDATA[15:0] in DFSDM_CHyWDATR register. That analog watchdog filter data is converted continuously (if CHEN=1 in DFSDM_CHyCFGR1 register) with the data rate given by the analog watchdog filter setting and the channel input clock frequency.

The analog watchdog filter conversion works like a regular Fast Continuous Conversion without the intergator. The number of serial samples needed for one result from analog watchdog filter output (at channel input clock frequency f_{CKIN}):

first conversion:

for $Sinc^x$ filters (x=1..5): number of samples = [F_{OSR} * F_{ORD} + F_{ORD} + 1]

for FastSinc filter: number of samples = [F_{OSR} * 4 + 2 + 1]

next conversions:

for $Sinc^x$ and FastSinc filters: number of samples = [FOSR * IOSR]

where:

F_{OSR} ..... filter oversampling ratio: F_{OSR} = AWFOSR[4:0]+1 (see DFSDM_CHyAWSCDR register)

F_{ORD} ..... the filter order: F_{ORD} = AWFORD[1:0] (see DFSDM_CHyAWSCDR register)

In case of output data register monitoring (AWFSEL=0), the comparison is done after a right bit shift and an offset correction of final data (see OFFSET[23:0] and DTRBS[4:0] fields in DFSDM_CHyCFGR2 register). A comparison is performed after each injected or regular end of conversion for the channels selected by AWDCH[3:0] field (in DFSDM_FLTxCR2 register).

The status of an analog watchdog event is signalized in DFSDM_FLTxAWSR register where a given event is latched. AWHTF[y]=1 flag signalizes crossing AWHT[23:0] value on channel y. AWLTF[y]=1 flag signalizes crossing AWLT[23:0] value on channel y. Latched events in DFSDM_FLTxAWSR register are cleared by writing ‘1’ into the corresponding clearing bit CLRAWHTF[y] or CLRAWLTF[y] in DFSDM_FLTxAWCFR register.

The global status of an analog watchdog is signalized by the AWDF flag bit in DFSDM_FLTxISR register (it is used for the fast detection of an interrupt source). AWDF=1 signalizes that at least one watchdog occurred (AWHTF[y]=1 or AWLTF[y]=1 for at least one channel). AWDF bit is cleared when all AWHTF[3:0] and AWLTF[3:0] are cleared.

An analog watchdog event can be assigned to break output signal. There are four break outputs to be assigned to a high or low threshold crossing event (dfsdm_break[3:0]). The break signal assignment to a given analog watchdog event is done by BKAWH[3:0] and BKAWL[3:0] fields in DFSDM_FLTxAWHTR and DFSDM_FLTxAWLTR register.

26.4.11 Short-circuit detector

The purpose of a short-circuit detector is to signalize with a very fast response time if an analog signal reached saturated values (out of full scale ranges) and remained on this value given time. This behavior can detect short-circuit or open circuit errors (e.g. overcurrent or overvoltage). An interrupt/event/break generation can be invoked.

Input data into a short-circuit detector is taken from channel transceiver outputs.

There is an upcounting counter on each input channel which is counting consecutive 0’s or 1’s on serial data receiver outputs. A counter is restarted if there is a change in the data stream received - 1 to 0 or 0 to 1 change of data signal. If this counter reaches a short-circuit threshold register value (SCDT[7:0] bits in DFSDM_CHyAWSCDR register), then a short-

circuit event is invoked. Each input channel has its short-circuit detector. Any channel can be selected to be continuously monitored by setting the SCDEN bit (in DFSDM_CHyCFGR1 register) and it has its own short-circuit detector settings (threshold value in SCDT[7:0] bits, status bit SCDF[3:0], status clearing bits CLRSCDF[3:0]). Status flag SCDF[y] is cleared also by hardware when corresponding channel y is disabled (CHEN[y] = 0).

On each channel, a short-circuit detector event can be assigned to break output signal dfsdm_break[3:0]. There are four break outputs to be assigned to a short-circuit detector event. The break signal assignment to a given channel short-circuit detector event is done by BKSCD[3:0] field in DFSDM_CHyAWSCDR register.

Short circuit detector cannot be used in case of parallel input data channel selection (DATMPX[1:0] ≠ 0 in DFSDM_CHyCFGR1 register).

Four break outputs are totally available (shared with the analog watchdog function).

26.4.12 Extreme detector

The purpose of an extremes detector is to collect the minimum and maximum values of final output data words (peak to peak values).

If the output data word is higher than the value stored in the extremes detector maximum register (EXMAX[23:0] bits in DFSDM_FLTxEXMAX register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMAXCH[1:0] bits (in DFSDM_FLTxEXMAX register) .

If the output data word is lower than the value stored in the extremes detector minimum register (EXMIN[23:0] bits in DFSDM_FLTxEXMIN register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMINCH[1:0] bits (in DFSDM_FLTxEXMIN register).

The minimum and maximum register values can be refreshed by software (by reading given DFSDM_FLTxEXMAX or DFSDM_FLTxEXMIN register). After refresh, the extremes detector minimum data register DFSDM_FLTxEXMIN is filled with 0x7FFFFF (maximum positive value) and the extremes detector maximum register DFSDM_FLTxEXMAX is filled with 0x800000 (minimum negative value).

The extremes detector performs a comparison after a right bit shift and an offset data correction. For each extremes detector, the input channels to be considered into computing the extremes value are selected in EXCH[3:0] bits (in DFSDM_FLTxCR2 register).

26.4.13 Data unit block

The data unit block is the last block of the whole processing path: External ΣΔ modulators - Serial transceivers - Sinc filter - Integrator - Data unit block.

The output data rate depends on the serial data stream rate, and filter and integrator settings. The maximum output data rate is:

$$Datarate[samples/s] = \frac{f_{CKIN}}{F_{OSR} \cdot (I_{OSR} - 1 + F_{ORD}) + (F_{ORD} + 1)} \quad \text{...FAST = 0, Sincx filter}$$

$$Datarate[samples/s] = \frac{f_{CKIN}}{F_{OSR} \cdot (I_{OSR} - 1 + 4) + (2 + 1)} \quad \text{...FAST = 0, FastSinc filter}$$

or

$$Datarate[samples/s] = \frac{f_{CKIN}}{F_{OSR} \cdot I_{OSR}} \quad \text{...FAST = 1}$$

Maximum output data rate in case of parallel data input:

$$Datarate[samples/s] = \frac{f_{DATAIN\_RATE}}{F_{OSR} \cdot (I_{OSR} - 1 + F_{ORD}) + (F_{ORD} + 1)} \quad \text{...FAST = 0, Sincx filter}$$

or

$$Datarate[samples/s] = \frac{f_{DATAIN\_RATE}}{F_{OSR} \cdot (I_{OSR} - 1 + 4) + (2 + 1)} \quad \text{...FAST = 0, FastSinc filter}$$

or

$$Datarate[samples/s] = \frac{f_{DATAIN\_RATE}}{F_{OSR} \cdot I_{OSR}} \quad \text{...FAST=1 or any filter bypass case } (F_{OSR} = 1)$$

where: f_{DATAIN_RATE}...input data rate from ADC or from CPU/DMA

The right bit-shift of final data is performed in this module because the final data width is 24-bit and data coming from the processing path can be up to 32 bits. This right bit-shift is configurable in the range 0-31 bits for each selected input channel (see DTRBS[4:0] bits in DFSDM_CHyCFGR2 register). The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained - to have valid 24-bit signed format of result data.

In the next step, an offset correction of the result is performed. The offset correction value (OFFSET[23:0] stored in register DFSDM_CHyCFGR2) is subtracted from the output data for a given channel. Data in the OFFSET[23:0] field is set by software by the appropriate calibration routine.

Due to the fact that all operations in digital processing are performed on 32-bit signed registers, the following conditions must be fulfilled not to overflow the result:

$FOSR^{FORD} \cdot IOSR <= 2^{31}$ ... for $Sinc^x$ filters, x = 1..5)
$2 \cdot FOSR^2 \cdot IOSR <= 2^{31}$ ... for FastSinc filter)

Note: In case of filter and integrator bypass (IOSR[7:0]=0, FOSR[9:0]=0), the input data rate (f_{DATAIN_RATE}) must be limited to be able to read all output data:
$f_{DATAIN\_RATE} \leq f_{APB}$
where f_{APB} is the bus frequency to which the DFSDM peripheral is connected.

26.4.14 Signed data format

Each DFSDM input serial channel can be connected to one external ΣΔ modulator. An external ΣΔ modulator can have 2 differential inputs (positive and negative) which can be used for a differential or single-ended signal measurement.

A ΣΔ modulator output is always assumed in a signed format (a data stream of zeros and ones from a ΣΔ modulator represents values -1 and +1).

Signed data format in registers: Data is in a signed format in registers for final output data, analog watchdog, extremes detector, offset correction. The msb of output data word represents the sign of value (two's complement format).

26.4.15 Launching conversions

Injected conversions can be launched using the following methods:

- Software: writing '1' to JSWSTART in the DFSDM_FLTxCR1 register.
- Trigger: JEXTSEL[4:0] selects the trigger signal while JEXTEN activates the trigger and selects the active edge at the same time (see the DFSDM_FLTxCR1 register).
- Synchronous with DFSDM_FLT0 if JSYNC=1: for DFSDM_FLTx (x>0), an injected conversion is automatically launched when in DFSDM_FLT0; the injected conversion is started by software (JSWSTART=1 in DFSDM_FLT0CR2 register). Each injected conversion in DFSDM_FLTx (x>0) is always executed according to its local configuration settings (JSCAN, JCHG, etc.).

If the scan conversion is enabled (bit JSCAN=1) then, each time an injected conversion is triggered, all of the selected channels in the injected group (JCHG[3:0] bits in DFSDM_FLTxJCHGR register) are converted sequentially, starting with the lowest channel (channel 0, if selected).

If the scan conversion is disabled (bit JSCAN=0) then, each time an injected conversion is triggered, only one of the selected channels in the injected group (JCHG[3:0] bits in DFSDM_FLTxJCHGR register) is converted and the channel selection is then moved to the next selected channel. Writing to the JCHG[3:0] bits when JSCAN=0 sets the channel selection to the lowest selected injected channel.

Only one injected conversion can be ongoing at a given time. Thus, any request to launch an injected conversion is ignored if another request for an injected conversion has already been issued but not yet completed.

Regular conversions can be launched using the following methods:

- Software: by writing '1' to RSWSTART in the DFSDM_FLTxCR1 register.
- Synchronous with DFSDM_FLT0 if RSYNC=1: for DFSDM_FLTx (x>0), a regular conversion is automatically launched when in DFSDM_FLT0; a regular conversion is started by software (RSWSTART=1 in DFSDM_FLT0CR2 register). Each regular conversion in DFSDM_FLTx (x>0) is always executed according to its local configuration settings (RCONT, RCH, etc.).

Only one regular conversion can be pending or ongoing at a given time. Thus, any request to launch a regular conversion is ignored if another request for a regular conversion has already been issued but not yet completed. A regular conversion can be pending if it was interrupted by an injected conversion or if it was started while an injected conversion was in progress. This pending regular conversion is then delayed and is performed when all injected conversion are finished. Any delayed regular conversion is signalized by RPEND bit in DFSDM_FLTxRDATAR register.

26.4.16 Continuous and fast continuous modes

Setting RCONT in the DFSDM_FLTxCR1 register causes regular conversions to execute in continuous mode. RCONT=1 means that the channel selected by RCH[1:0] is converted repeatedly after '1' is written to RSWSTART.

The regular conversions executing in continuous mode can be stopped by writing ‘0’ to RCONT. After clearing RCONT, the on-going conversion is stopped immediately.

In continuous mode, the data rate can be increased by setting the FAST bit in the DFSDM_FLTxCR1 register. In this case, the filter does not need to be refilled by new fresh data if converting continuously from one channel because data inside the filter is valid from previously sampled continuous data. The speed increase depends on the chosen filter order. The first conversion in fast mode (FAST=1) after starting a continuous conversion by RSWSTART=1 takes still full time (as when FAST=0), then each subsequent conversion is finished in shorter intervals.

Conversion time in continuous mode:

if FAST = 0 (or first conversion if FAST=1):

for $Sinc^x$ filters:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$

for FastSinc filter:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + 4) + 2] / f_{CKIN}$

if FAST = 1 (except first conversion):

for $Sinc^x$ and FastSinc filters:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, only integrator active):

$t = I_{OSR} / f_{CKIN}$ (... but CNVCNT=0)

Continuous mode is not available for injected conversions. Injected conversions can be started by timer trigger to emulate the continuous mode with precise timing.

If a regular continuous conversion is in progress (RCONT=1) and if a write access to DFSDM_FLTxCR1 register requesting regular continuous conversion (RCONT=1) is performed, then regular continuous conversion is restarted from the next conversion cycle (like new regular continuous conversion is applied for new channel selection - even if there is no change in DFSDM_FLTxCR1 register).

26.4.17 Request precedence

An injected conversion has a higher precedence than a regular conversion. A regular conversion which is already in progress is immediately interrupted by the request of an injected conversion; this regular conversion is restarted after the injected conversion finishes.

An injected conversion cannot be launched if another injected conversion is pending or already in progress: any request to launch an injected conversion (either by JSWSTART or by a trigger) is ignored as long as bit JCIP is ‘1’ (in the DFSDM_FLTxISR register).

Similarly, a regular conversion cannot be launched if another regular conversion is pending or already in progress: any request to launch a regular conversion (using RSWSTART) is ignored as long as bit RCIP is ‘1’ (in the DFSDM_FLTxISR register).

However, if an injected conversion is requested while a regular conversion is already in progress, the regular conversion is immediately stopped and an injected conversion is launched. The regular conversion is then restarted and this delayed restart is signalized in bit RPEND.

Injected conversions have precedence over regular conversions in that a injected conversion can temporarily interrupt a sequence of continuous regular conversions. When

the sequence of injected conversions finishes, the continuous regular conversions start again if RCONT is still set (and RPEND bit will signalize the delayed start on the first regular conversion result).

Precedence also matters when actions are initiated by the same write to DFSDM, or if multiple actions are pending at the end of another action. For example, suppose that, while an injected conversion is in process (JCIP=1), a single write operation to DFSDM_FLTxCR1 writes ‘1’ to RSWSTART, requesting a regular conversion. When the injected sequence finishes, the precedence dictates that the regular conversion is performed next and its delayed start is signalized in RPEND bit.

26.4.18 Power optimization in run mode

In order to reduce the consumption, the DFSDM filter and integrator are automatically put into idle when not used by conversions (RCIP=0, JCIP=0).

26.5 DFSDM interrupts

In order to increase the CPU performance, a set of interrupts related to the CPU event occurrence has been implemented:

- End of injected conversion interrupt:
  - enabled by JEOCIE bit in DFSDM_FLTxCR2 register
  - indicated in JEOCF bit in DFSDM_FLTxISR register
  - cleared by reading DFSDM_FLTxJDATAR register (injected data)
  - indication of which channel end of conversion occurred, reported in JDATACH[1:0] bits in DFSDM_FLTxJDATAR register
- End of regular conversion interrupt:
  - enabled by REOCIE bit in DFSDM_FLTxCR2 register
  - indicated in REOCF bit in DFSDM_FLTxISR register
  - cleared by reading DFSDM_FLTxRDATAR register (regular data)
  - indication of which channel end of conversion occurred, reported in RDATACH[1:0] bits in DFSDM_FLTxRDATAR register
- Data overrun interrupt for injected conversions:
  - occurred when injected converted data were not read from DFSDM_FLTxJDATAR register (by CPU or DMA) and were overwritten by a new injected conversion
  - enabled by JOVRIE bit in DFSDM_FLTxCR2 register
  - indicated in JOVRF bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRJOVRF bit in DFSDM_FLTxICR register
- Data overrun interrupt for regular conversions:
  - occurred when regular converted data were not read from DFSDM_FLTxRDATAR register (by CPU or DMA) and were overwritten by a new regular conversion
  - enabled by ROVRIE bit in DFSDM_FLTxCR2 register
  - indicated in ROVRF bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRROVRF bit in DFSDM_FLTxICR register
- Analog watchdog interrupt:

  - occurred when converted data (output data or data from analog watchdog filter - according to AWFSEL bit setting in DFSDM_FLTxCR1 register) crosses over/under high/low thresholds in DFSDM_FLTxAWHTR / DFSDM_FLTxAWLTR registers
  - enabled by AWDIE bit in DFSDM_FLTxCR2 register (on selected channels AWDCH[3:0])
  - indicated in AWDF bit in DFSDM_FLTxISR register
  - separate indication of high or low analog watchdog threshold error by AWHTF[3:0] and AWLTF[3:0] fields in DFSDM_FLTxAWSR register
  - cleared by writing ‘1’ into corresponding CLRAWHTF[3:0] or CLRAWLTF[3:0] bits in DFSDM_FLTxAWCFR register
- Short-circuit detector interrupt:
  - occurred when the number of stable data crosses over thresholds in DFSDM_CHyAWSCDR register
  - enabled by SCDIE bit in DFSDM_FLTxCR2 register (on channel selected by SCDEN bi tin DFSDM_CHyCFGR1 register)
  - indicated in SCDF[3:0] bits in DFSDM_FLTxISR register (which also reports the channel on which the short-circuit detector event occurred)
  - cleared by writing ‘1’ into the corresponding CLRSCDF[3:0] bit in DFSDM_FLTxICR register
- Channel clock absence interrupt:
  - occurred when there is clock absence on CKINy pin (see *Clock absence detection* in *Section 26.4.4: Serial channel transceivers*)
  - enabled by CKABIE bit in DFSDM_FLTxCR2 register (on channels selected by CKABEN bit in DFSDM_CHyCFGR1 register)
  - indicated in CKABF[y] bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRCKABF[y] bit in DFSDM_FLTxICR register

Table 204. DFSDM interrupt requests

Interrupt event	Event flag	Event/Interrupt clearing method	Interrupt enable control bit
End of injected conversion	JEOCF	reading DFSDM_FLTxJDATAR	JEOCIE
End of regular conversion	REOCF	reading DFSDM_FLTxRDATAR	REOCIE
Injected data overrun	JOVRF	writing CLRJOVRF = 1	JOVRIE
Regular data overrun	ROVRF	writing CLRROVRF = 1	ROVRIE
Analog watchdog	AWDF, AWHTF[3:0], AWLTF[3:0]	writing CLRAWHTF[3:0] = 1 writing CLRAWLTF[3:0] = 1	AWDIE, (AWDCH[3:0])
short-circuit detector	SCDF[3:0]	writing CLRSCDF[3:0] = 1	SCDIE, (SCDEN)
Channel clock absence	CKABF[3:0]	writing CLRCKABF[3:0] = 1	CKABIE, (CKABEN)

26.6 DFSDM DMA transfer

To decrease the CPU intervention, conversions can be transferred into memory using a DMA transfer. A DMA transfer for injected conversions is enabled by setting bit JDMAEN=1 in DFSDM_FLTxCR1 register. A DMA transfer for regular conversions is enabled by setting bit RDMAEN=1 in DFSDM_FLTxCR1 register.

Note: With a DMA transfer, the interrupt flag is automatically cleared at the end of the injected or regular conversion (JEOCF or REOCF bit in DFSDM_FLTxISR register) because DMA is reading DFSDM_FLTxJDATAR or DFSDM_FLTxRDATAR register.

26.7 DFSDM channel y registers (y=0..3)

Word access (32-bit) must be used for registers write access except DFSDM_CHyDATINR register. Write access to DFSDM_CHyDATINR register can be either word access (32-bit) or half-word access (16-bit).

26.7.1 DFSDM channel y configuration register (DFSDM_CHyCFGR1)

This register specifies the parameters used by channel y.

Address offset: 0x00 + 0x20 * y, (y = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFSDMEN	CKOUTSRC	Res.	Res.	Res.	Res.	Res.	Res.	CKOUTDIV[7:0]							
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
DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
rw	rw	rw	rw				rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **DFSDMEN**: Global enable for DFSDM interface

0: DFSDM interface disabled

1: DFSDM interface enabled

If DFSDM interface is enabled, then it is started to operate according to enabled y channels and enabled x filters settings (CHEN bit in DFSDM_CHyCFGR1 and DFEN bit in DFSDM_FLTxCR1). Data cleared by setting DFSDMEN=0:

–all registers DFSDM_FLTxISR are set to reset state (x = 0..3)

–all registers DFSDM_FLTxAWSR are set to reset state (x = 0..3)

Note: DFSDMEN is present only in DFSDM_CH0CFGR1 register (channel y=0)

Bit 30 **CKOUTSRC**: Output serial clock source selection

0: Source for output clock is from system clock

1: Source for output clock is from audio clock

This value can be modified only when DFSDMEN=0 (in DFSDM_CH0CFGR1 register).

Note: CKOUTSRC is present only in DFSDM_CH0CFGR1 register (channel y=0)

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **CKOUTDIV[7:0]**: Output serial clock divider

0: Output clock generation is disabled (CKOUT signal is set to low state)

1- 255: Defines the division of system clock for the serial clock output for CKOUT signal in range 2 - 256 (Divider = CKOUTDIV+1).

CKOUTDIV also defines the threshold for a clock absence detection.

This value can only be modified when DFSDMEN=0 (in DFSDM_CH0CFGR1 register).

If DFSDMEN=0 (in DFSDM_CH0CFGR1 register) then CKOUT signal is set to low state (setting is performed one DFSDM clock cycle after DFSDMEN=0).

Note: CKOUTDIV is present only in DFSDM_CH0CFGR1 register (channel y=0)

Bits 15:14 **DATPACK[1:0]**: Data packing mode in DFSDM_CHyDATINR register.

0: Standard: input data in DFSDM_CHyDATINR register are stored only in INDAT0[15:0]. To empty DFSDM_CHyDATINR register one sample must be read by the DFSDM filter from channel y.

1: Interleaved: input data in DFSDM_CHyDATINR register are stored as two samples:

–first sample in INDAT0[15:0] (assigned to channel y)

–second sample INDAT1[15:0] (assigned to channel y)

To empty DFSDM_CHyDATINR register, two samples must be read by the digital filter from channel y (INDAT0[15:0] part is read as first sample and then INDAT1[15:0] part is read as next sample).

2: Dual: input data in DFSDM_CHyDATINR register are stored as two samples:

–first sample INDAT0[15:0] (assigned to channel y)

–second sample INDAT1[15:0] (assigned to channel y+1)

To empty DFSDM_CHyDATINR register first sample must be read by the digital filter from channel y and second sample must be read by another digital filter from channel y+1. Dual mode is available only on even channel numbers (y = 0, 2), for odd channel numbers (y = 1, 3) DFSDM_CHyDATINR is write protected. If an even channel is set to dual mode then the following odd channel must be set into standard mode (DATPACK[1:0]=0) for correct cooperation with even channel.

3: Reserved

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 13:12 **DATMPX[1:0]**: Input data multiplexer for channel y

0: Data to channel y are taken from external serial inputs as 1-bit values. DFSDM_CHyDATINR register is write protected.

1: Data to channel y are taken from internal analog to digital converter ADC_{y+1} output register update as 16-bit values (if ADC_{y+1} is available). Data from ADCs are written into INDAT0[15:0] part of DFSDM_CHyDATINR register.

2: Data to channel y are taken from internal DFSDM_CHyDATINR register by direct CPU/DMA write. There can be written one or two 16-bit data samples according DATPACK[1:0] bit field setting.

3: Reserved

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **CHINSEL**: Channel inputs selection

0: Channel inputs are taken from pins of the same channel y.

1: Channel inputs are taken from pins of the following channel (channel (y+1) modulo 8).

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bit 7 **CHEN**: Channel y enable

0: Channel y disabled

1: Channel y enabled

If channel y is enabled, then serial data receiving is started according to the given channel setting.

Bit 6 **CKABEN**: Clock absence detector enable on channel y
0: Clock absence detector disabled on channel y
1: Clock absence detector enabled on channel y

Bit 5 **SCDEN**: Short-circuit detector enable on channel y
0: Input channel y will not be guarded by the short-circuit detector
1: Input channel y will be continuously guarded by the short-circuit detector

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **SPICKSEL[1:0]**: SPI clock select for channel y
0: clock coming from external CKINy input - sampling point according SITP[1:0]
1: clock coming from internal CKOUT output - sampling point according SITP[1:0]
2: clock coming from internal CKOUT - sampling point on each second CKOUT falling edge. For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
3: clock coming from internal CKOUT output - sampling point on each second CKOUT rising edge. For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).
This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 1:0 **SITP[1:0]**: Serial interface type for channel y
00: SPI with rising edge to strobe data
01: SPI with falling edge to strobe data
10: Manchester coded input on DATINy pin: rising edge = logic 0, falling edge = logic 1
11: Manchester coded input on DATINy pin: rising edge = logic 1, falling edge = logic 0
This value can only be modified when CHEN=0 (in DFSDM_CHyCFGR1 register).

26.7.2 DFSDM channel y configuration register (DFSDM_CHyCFGR2)

This register specifies the parameters used by channel y.

Address offset: 0x04 + 0x20 * y, (y = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET[23:8]															
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
OFFSET[7:0]								DTRBS[4:0]					Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:8 **OFFSET[23:0]**: 24-bit calibration offset for channel y
For channel y, OFFSET is applied to the results of each conversion from this channel.
This value is set by software.

Bits 7:3 **DTRBS[4:0]**: Data right bit-shift for channel y
0-31: Defines the shift of the data result coming from the integrator - how many bit shifts to the right will be performed to have final results. Bit-shift is performed before offset correction. The data shift is rounding the result to nearest integer value. The sign of shifted result is maintained (to have valid 24-bit signed format of result data).
This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 2:0 Reserved, must be kept at reset value.

26.7.3 DFSDM channel y analog watchdog and short-circuit detector register (DFSDM_CHyAWSCDR)

Short-circuit detector and analog watchdog settings for channel y.

Address offset: 0x08 + 0x20 * y, (y = 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.	AWFORD[1:0]		Res.	AWFOSR[4:0]				
								rw	rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
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 **AWFORD[1:0]**: Analog watchdog Sinc filter order on channel y
0: FastSinc filter type
1: $Sinc^1$ filter type
2: $Sinc^2$ filter type
3: $Sinc^3$ filter type
$Sinc^x$ filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^x$$

FastSinc filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

This bit can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **AWFOSR[4:0]**: Analog watchdog filter oversampling ratio (decimation rate) on channel y
0 - 31: Defines the length of the Sinc type filter in the range 1 - 32 (AWFOSR + 1). This number is also the decimation ratio of the analog data rate.
This bit can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).
Note: If AWFOSR = 0 then the filter has no effect (filter bypass).

Bits 15:12 **BKSCD[3:0]**: Break signal assignment for short-circuit detector on channel y
BKSCD[i] = 0: Break i signal not assigned to short-circuit detector on channel y
BKSCD[i] = 1: Break i signal assigned to short-circuit detector on channel y

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:0 **SCDT[7:0]**: short-circuit detector threshold for channel y
These bits are written by software to define the threshold counter for the short-circuit detector. If this value is reached, then a short-circuit detector event occurs on a given channel.

26.7.4 DFSDM channel y watchdog filter data register (DFSDM_CHyWDATR)

This register contains the data resulting from the analog watchdog filter associated to the input channel y.

Address offset: 0x0C + 0x20 * y, (y = 0 to 3)

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDATA[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 **WDATA[15:0]**: Input channel y watchdog data
Data converted by the analog watchdog filter for input channel y. This data is continuously converted (no trigger) for this channel, with a limited resolution (OSR=1..32/sinc order = 1..3).

26.7.5 DFSDM channel y data input register (DFSDM_CHyDATINR)

This register contains 16-bit input data to be processed by DFSDM filter module. Write access can be either word access (32-bit) or half-word access (16-bit).

Address offset: 0x10 + 0x20 * y, (y = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INDAT1[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
INDAT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **INDAT1[15:0]**: Input data for channel y or channel y+1

Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2. Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).

If DATPACK[1:0]=0 (standard mode)

INDAT0[15:0] is write protected (not used for input sample).

If DATPACK[1:0]=1 (interleaved mode)

Second channel y data sample is stored into INDAT1[15:0]. First channel y data sample is stored into INDAT0[15:0]. Both samples are read sequentially by DFSDM_FLTx filter as two channel y data samples.

If DATPACK[1:0]=2 (dual mode).

For even y channels: sample in INDAT1[15:0] is automatically copied into INDAT0[15:0] of channel (y+1).

For odd y channels: INDAT1[15:0] is write protected.

See *Section 26.4.6: Parallel data inputs* for more details.

INDAT0[15:1] is in the16-bit signed format.

Bits 15:0 **INDAT0[15:0]**: Input data for channel y

Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2. Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).

If DATPACK[1:0]=0 (standard mode)

Channel y data sample is stored into INDAT0[15:0].

If DATPACK[1:0]=1 (interleaved mode)

First channel y data sample is stored into INDAT0[15:0]. Second channel y data sample is stored into INDAT1[15:0]. Both samples are read sequentially by DFSDM_FLTx filter as two channel y data samples.

If DATPACK[1:0]=2 (dual mode).

For even y channels: Channel y data sample is stored into INDAT0[15:0].

For odd y channels: INDAT0[15:0] is write protected.

See *Section 26.4.6: Parallel data inputs* for more details.

INDAT0[15:0] is in the16-bit signed format.

26.7.6 DFSDM channel y delay register (DFSDM_CHyDLYR)

Address offset: 0x14 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	PLSSKP[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **PLSSKP[5:0]**: Pulses to skip for input data skipping function

0-63: Defines the number of serial input samples that will be skipped. Skipping is applied immediately after writing to this field. Reading of PLSSKP[5:0] returns current value of pulses which will be skipped. If PLSSKP[5:0]=0 then all required data samples were already skipped.

Note: User can update PLSSKP[5:0] also when PLSSKP[5:0] is not zero.

26.8 DFSDM filter x module registers (x=0..3)

Word access (32-bit) must be used for registers write access except DFSDM_CHyDATINR register.

26.8.1 DFSDM filter x control register 1 (DFSDM_FLTxCR1)

Address offset: 0x100 + 0x80 * 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.	AWFSEL	FAST	Res.	Res.	Res.	RCH[1:0]		Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSWSTART	Res.
	rw	rw				rw	rw			rw		rw	rw	rt_w1	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSWSTART	DFEN
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw		rt_w1	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **AWFSEL**: Analog watchdog fast mode select

0: Analog watchdog on data output value (after the digital filter). The comparison is done after offset correction and shift

1: Analog watchdog on channel transceivers value (after watchdog filter)

Bit 29 **FAST**: Fast conversion mode selection for regular conversions

0: Fast conversion mode disabled

1: Fast conversion mode enabled

When converting a regular conversion in continuous mode, having enabled the fast mode causes each conversion (except the first) to execute faster than in standard mode. This bit has no effect on conversions which are not continuous.

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

if FAST=0 (or first conversion in continuous mode if FAST=1):

$t = [F_{OSR} * (I_{OSR}\text{-}1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$..... for $Sinc^x$ filters

$t = [F_{OSR} * (I_{OSR}\text{-}1 + 4) + 2] / f_{CKIN}$..... for FastSinc filter

if FAST=1 in continuous mode (except first conversion):

$t = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case if $F_{OSR} = F_{OSR}[9:0]+1 = 1$ (filter bypassed, active only integrator):

$t = I_{OSR} / f_{CKIN}$ (... but CNVCNT=0)

where: f_{CKIN} is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input.

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:24 **RCH[1:0]**: Regular channel selection
0: Channel 0 is selected as the regular channel
1: Channel 1 is selected as the regular channel
...
3: Chanel 3 is selected as the regular channel
Writing this bit when RCIP=1 takes effect when the next regular conversion begins. This is especially useful in continuous mode (when RCONT=1). It also affects regular conversions which are pending (due to ongoing injected conversion).

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RDMAEN**: DMA channel enabled to read data for the regular conversion
0: The DMA channel is not enabled to read regular data
1: The DMA channel is enabled to read regular data
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RSYNC**: Launch regular conversion synchronously with DFSDM_FLT0
0: Do not launch a regular conversion synchronously with DFSDM_FLT0
1: Launch a regular conversion in this DFSDM_FLTx at the very moment when a regular conversion is launched in DFSDM_FLT0
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 18 **RCONT**: Continuous mode selection for regular conversions
0: The regular channel is converted just once for each conversion request
1: The regular channel is converted repeatedly after each conversion request
Writing ‘0’ to this bit while a continuous regular conversion is already in progress stops the continuous mode immediately.

Bit 17 **RSWSTART**: Software start of a conversion on the regular channel
0: Writing ‘0’ has no effect
1: Writing ‘1’ makes a request to start a conversion on the regular channel and causes RCIP to become ‘1’. If RCIP=1 already, writing to RSWSTART has no effect. Writing ‘1’ has no effect if RSYNC=1.
This bit is always read as ‘0’.

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:13 **JEXTEN[1:0]**: Trigger enable and trigger edge selection for injected conversions
00: Trigger detection is disabled
01: Each rising edge on the selected trigger makes a request to launch an injected conversion
10: Each falling edge on the selected trigger makes a request to launch an injected conversion
11: Both rising edges and falling edges on the selected trigger make requests to launch injected conversions
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bits 12:8 **JEXTSEL[4:0]**: Trigger signal selection for launching injected conversions

0x0-0x1F: Trigger inputs selected by the following table (internal or external trigger).

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Note: synchronous trigger has latency up to one $f_{DFSDMCLK}$ clock cycle (with deterministic jitter), asynchronous trigger has latency 2-3 $f_{DFSDMCLK}$ clock cycles (with jitter up to 1 cycle).

	DFSDM_FLTx
0x00	dfsdm_jtrg0
0x01	dfsdm_jtrg1
...	
0x1E	dfsdm_jtrg30
0x1F	dfsdm_jtrg31

Refer to *Table 200: DFSDM triggers connection*.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **JDMAEN**: DMA channel enabled to read data for the injected channel group

0: The DMA channel is not enabled to read injected data

1: The DMA channel is enabled to read injected data

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 4 **JSCAN**: Scanning conversion mode for injected conversions

0: One channel conversion is performed from the injected channel group and next the selected channel from this group is selected.

1: The series of conversions for the injected group channels is executed, starting over with the lowest selected channel.

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Writing JCHG if JSCAN=0 resets the channel selection to the lowest selected channel.

Bit 3 **JSYNC**: Launch an injected conversion synchronously with the DFSDM_FLT0 JSWSTART trigger

0: Do not launch an injected conversion synchronously with DFSDM_FLT0

1: Launch an injected conversion in this DFSDM_FLTx at the very moment when an injected conversion is launched in DFSDM_FLT0 by its JSWSTART trigger

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **JSWSTART**: Start a conversion of the injected group of channels

0: Writing ‘0’ has no effect.

1: Writing ‘1’ makes a request to convert the channels in the injected conversion group, causing JCIP to become ‘1’ at the same time. If JCIP=1 already, then writing to JSWSTART has no effect. Writing ‘1’ has no effect if JSYNC=1.

This bit is always read as ‘0’.

Bit 0 **DFEN**: DFSDM_FLTx enable

0: DFSDM_FLTx is disabled. All conversions of given DFSDM_FLTx are stopped immediately and all DFSDM_FLTx functions are stopped.

1: DFSDM_FLTx is enabled. If DFSDM_FLTx is enabled, then DFSDM_FLTx starts operating according to its setting.

Data which are cleared by setting DFEN=0:

–register DFSDM_FLTxISR is set to the reset state

–register DFSDM_FLTxAWSR is set to the reset state

26.8.2 DFSDM filter x control register 2 (DFSDM_FLTxCR2)

Address offset: 0x104 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[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.	EXCH[3:0]				Res.	CKABIE	SCDIE	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

Bits 19:16 **AWDCH[3:0]**: Analog watchdog channel selection
These bits select the input channel to be guarded continuously by the analog watchdog.
AWDCH[y] = 0: Analog watchdog is disabled on channel y
AWDCH[y] = 1: Analog watchdog is enabled on channel y

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **EXCH[3:0]**: Extremes detector channel selection
These bits select the input channels to be taken by the Extremes detector.
EXCH[y] = 0: Extremes detector does not accept data from channel y
EXCH[y] = 1: Extremes detector accepts data from channel y

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CKABIE**: Clock absence interrupt enable
0: Detection of channel input clock absence interrupt is disabled
1: Detection of channel input clock absence interrupt is enabled
Please see the explanation of CKABF[3:0] in DFSDM_FLTxISR.
Note: CKABIE is present only in DFSDM_FLT0CR2 register (filter x=0)

Bit 5 **SCDIE**: Short-circuit detector interrupt enable
0: short-circuit detector interrupt is disabled
1: short-circuit detector interrupt is enabled
Please see the explanation of SCDF[3:0] in DFSDM_FLTxISR.
Note: SCDIE is present only in DFSDM_FLT0CR2 register (filter x=0)

Bit 4 **AWDIE**: Analog watchdog interrupt enable
0: Analog watchdog interrupt is disabled
1: Analog watchdog interrupt is enabled
Please see the explanation of AWDF in DFSDM_FLTxISR.

Bit 3 **ROVRIE**: Regular data overrun interrupt enable
0: Regular data overrun interrupt is disabled
1: Regular data overrun interrupt is enabled
Please see the explanation of ROVRF in DFSDM_FLTxISR.

Bit 2 **JOVRIE**: Injected data overrun interrupt enable
0: Injected data overrun interrupt is disabled
1: Injected data overrun interrupt is enabled
Please see the explanation of JOVRF in DFSDM_FLTxISR.

Bit 1 **REOCIE**: Regular end of conversion interrupt enable
0: Regular end of conversion interrupt is disabled
1: Regular end of conversion interrupt is enabled
Please see the explanation of REOCF in DFSDM_FLTxISR.

Bit 0 **JEOCIE**: Injected end of conversion interrupt enable
0: Injected end of conversion interrupt is disabled
1: Injected end of conversion interrupt is enabled
Please see the explanation of JEOCF in DFSDM_FLTxISR.

26.8.3 DFSDM filter x interrupt and status register (DFSDM_FLTxISR)

Address offset: 0x108 + 0x80 * x, (x = 0 to 3)

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SCDF[3:0]				Res.	Res.	Res.	Res.	CKABF[3:0]			
				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.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	r	r									r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **SCDF[3:0]**: short-circuit detector flag
SDCF[y]=0: No short-circuit detector event occurred on channel y
SDCF[y]=1: The short-circuit detector counter reaches, on channel y, the value programmed in the DFSDM_CHyAWSCDR registers
This bit is set by hardware. It can be cleared by software using the corresponding CLRSCDF[y] bit in the DFSDM_FLTxICR register. SCDF[y] is cleared also by hardware when CHEN[y] = 0 (given channel is disabled).
Note: SCDF[3:0] is present only in DFSDM_FLT0ISR register (filter x=0)

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CKABF[3:0]**: Clock absence flag
CKABF[y]=0: Clock signal on channel y is present.
CKABF[y]=1: Clock signal on channel y is not present.
Given y bit is set by hardware when clock absence is detected on channel y. It is held at CKABF[y]=1 state by hardware when CHEN=0 (see DFSDM_CHyCFGR1 register). It is held at CKABF[y]=1 state by hardware when the transceiver is not yet synchronized.It can be cleared by software using the corresponding CLRCKABF[y] bit in the DFSDM_FLTxICR register.
Note: CKABF[3:0] is present only in DFSDM_FLT0ISR register (filter x=0)

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RCIP**: Regular conversion in progress status
0: No request to convert the regular channel has been issued
1: The conversion of the regular channel is in progress or a request for a regular conversion is pending
A request to start a regular conversion is ignored when RCIP=1.

Bit 13 **JCIP**: Injected conversion in progress status
0: No request to convert the injected channel group (neither by software nor by trigger) has been issued
1: The conversion of the injected channel group is in progress or a request for a injected conversion is pending, due either to ‘1’ being written to JSWSTART or to a trigger detection
A request to start an injected conversion is ignored when JCIP=1.

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **AWDF**: Analog watchdog
0: No Analog watchdog event occurred
1: The analog watchdog block detected voltage which crosses the value programmed in the DFSDM_FLTxAWLTR or DFSDM_FLTxAWHTR registers.
This bit is set by hardware. It is cleared by software by clearing all source flag bits AWHTF[3:0] and AWLTF[3:0] in DFSDM_FLTxAWSR register (by writing ‘1’ into the clear bits in DFSDM_FLTxAWCFR register).

Bit 3 **ROVRF**: Regular conversion overrun flag
0: No regular conversion overrun has occurred
1: A regular conversion overrun has occurred, which means that a regular conversion finished while REOCF was already ‘1’. RDATAR is not affected by overruns
This bit is set by hardware. It can be cleared by software using the CLRROVRF bit in the DFSDM_FLTxICR register.

Bit 2 **JOVRF**: Injected conversion overrun flag
0: No injected conversion overrun has occurred
1: An injected conversion overrun has occurred, which means that an injected conversion finished while JEOCF was already ‘1’. JDATAR is not affected by overruns
This bit is set by hardware. It can be cleared by software using the CLRJOVRF bit in the DFSDM_FLTxICR register.

Bit 1 **REOCF**: End of regular conversion flag
0: No regular conversion has completed
1: A regular conversion has completed and its data may be read
This bit is set by hardware. It is cleared when the software or DMA reads DFSDM_FLTxRDATAR.

Bit 0 **JEOCF**: End of injected conversion flag
0: No injected conversion has completed
1: An injected conversion has completed and its data may be read
This bit is set by hardware. It is cleared when the software or DMA reads DFSDM_FLTxJDATAR.

Note: For each of the flag bits, an interrupt can be enabled by setting the corresponding bit in DFSDM_FLTxCR2. If an interrupt is called, the flag must be cleared before exiting the interrupt service routine.

All the bits of DFSDM_FLTxISR are automatically reset when DFEN=0.

26.8.4 DFSDM filter x interrupt flag clear register (DFSDM_FLTxICR)

Address offset: 0x10C + 0x80 * 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.	CLRSCDF[3:0]				Res.	Res.	Res.	Res.	CLRCKABF[3:0]			
				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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRROVRF	CLRJOVRF	Res.	Res.
												rc_w1	rc_w1		

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CLRSCDF[3:0]**: Clear the short-circuit detector flag
CLRSCDF[y]=0: Writing ‘0’ has no effect
CLRSCDF[y]=1: Writing ‘1’ to position y clears the corresponding SCDF[y] bit in the DFSDM_FLTxISR register
Note: CLRSCDF[3:0] is present only in DFSDM_FLT0ICR register (filter x=0)

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CLRCKABF[3:0]**: Clear the clock absence flag
CLRCKABF[y]=0: Writing ‘0’ has no effect
CLRCKABF[y]=1: Writing ‘1’ to position y clears the corresponding CKABF[y] bit in the DFSDM_FLTxISR register. When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y].
Note: CLRCKABF[3:0] is present only in DFSDM_FLT0ICR register (filter x=0)

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CLRROVRF**: Clear the regular conversion overrun flag
0: Writing ‘0’ has no effect
1: Writing ‘1’ clears the ROVRF bit in the DFSDM_FLTxISR register

Bit 2 **CLRJOVRF**: Clear the injected conversion overrun flag
0: Writing ‘0’ has no effect
1: Writing ‘1’ clears the JOVRF bit in the DFSDM_FLTxISR register

Bits 1:0 Reserved, must be kept at reset value.

Note: The bits of DFSDM_FLTxICR are always read as ‘0’.

26.8.5 DFSDM filter x injected channel group selection register (DFSDM_FLTxJCHGR)

Address offset: 0x110 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0001

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

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **JCHG[3:0]**: Injected channel group selection

JCHG[y]=0: channel y is not part of the injected group

JCHG[y]=1: channel y is part of the injected group

If JSCAN=1, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first and the sequence ends at the highest selected channel.

If JSCAN=0, then only one channel is converted from the selected channels, and the channel selection is moved to the next channel. Writing JCHG, if JSCAN=0, resets the channel selection to the lowest selected channel.

At least one channel must always be selected for the injected group. Writes causing all JCHG bits to be zero are ignored.

26.8.6 DFSDM filter x control register (DFSDM_FLTxFCR)

Address offset: 0x114 + 0x80 * 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
FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]									
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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **FORD[2:0]**: Sinc filter order

0: FastSinc filter type
1: $Sinc^1$ filter type
2: $Sinc^2$ filter type
3: $Sinc^3$ filter type
4: $Sinc^4$ filter type
5: $Sinc^5$ filter type
6-7: Reserved

$Sinc^x$ filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^x$$

FastSinc filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

This bit can only be modified when DFEN=0 (DFSDM_FLTxCR1).

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:16 **FOSR[9:0]**: Sinc filter oversampling ratio (decimation rate)

0 - 1023: Defines the length of the Sinc type filter in the range 1 - 1024 (F_{OSR} = FOSR[9:0] +1). This number is also the decimation ratio of the output data rate from filter.
This bit can only be modified when DFEN=0 (DFSDM_FLTxCR1)

Note: If FOSR = 0, then the filter has no effect (filter bypass).

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

Bits 7:0 **IOSR[7:0]**: Integrator oversampling ratio (averaging length)

0- 255: The length of the Integrator in the range 1 - 256 (IOSR + 1). Defines how many samples from Sinc filter will be summed into one output data sample from the integrator. The output data rate from the integrator will be decreased by this number (additional data decimation ratio).
This bit can only be modified when DFEN=0 (DFSDM_FLTxCR1)

Note: If IOSR = 0, then the Integrator has no effect (Integrator bypass).

26.8.7 DFSDM filter x data register for injected group (DFSDM_FLTxJDATAR)

Address offset: 0x118 + 0x80 * 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
JDATA[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	JDATACH[1:0]	
r	r	r	r	r	r	r	r							r	r

Bits 31:8 **JDATA[23:0]**: Injected group conversion data
When each conversion of a channel in the injected group finishes, its resulting data is stored in this field. The data is valid when JEOCF=1. Reading this register clears the corresponding JEOCF.

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

Bits 1:0 **JDATACH[1:0]**: Injected channel most recently converted
When each conversion of a channel in the injected group finishes, JDATACH[1:0] is updated to indicate which channel was converted. Thus, JDATA[23:0] holds the data that corresponds to the channel indicated by JDATACH[1:0].

Note: DMA may be used to read the data from this register. Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears JEOCF in DFSDM_FLTxISR. Thus, the firmware must not read this register if DMA is activated to read data from this register.

26.8.8 DFSDM filter x data register for the regular channel (DFSDM_FLTxRDATAR)

Address offset: 0x11C + 0x80 * 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
RDATA[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[7:0]								Res.	Res.	Res.	RPEND	Res.	Res.	RDATACH[1:0]	
r	r	r	r	r	r	r	r				r			r	r

Bits 31:8 **RDATA[23:0]**: Regular channel conversion data
When each regular conversion finishes, its data is stored in this register. The data is valid when REOCF=1. Reading this register clears the corresponding REOCF.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **RPEND**: Regular channel pending data
Regular data in RDATA[23:0] was delayed due to an injected channel trigger during the conversion

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **RDATACH[1:0]**: Regular channel most recently converted
When each regular conversion finishes, RDATACH[1:0] is updated to indicate which channel was converted (because regular channel selection RCH[1:0] in DFSDM_FLTxCR1 register can be updated during regular conversion). Thus RDATA[23:0] holds the data that corresponds to the channel indicated by RDATACH[1:0].

Note: Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears REOCF in DFSDM_FLTxISR.

26.8.9 DFSDM filter x analog watchdog high threshold register (DFSDM_FLTxAWHTR)

Address offset: 0x120 + 0x80 * 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
AWHT[23:8]															
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
AWHT[7:0]								Res.	Res.	Res.	Res.	BKAWH[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:8 **AWHT[23:0]**: Analog watchdog high threshold

These bits are written by software to define the high threshold for the analog watchdog.

Note: In case channel transceivers monitor (AWFSEL=1), the higher 16 bits (AWHT[23:8]) define the 16-bit threshold as compared with the analog watchdog filter output (because data coming from the analog watchdog filter are up to a 16-bit resolution). Bits AWHT[7:0] are not taken into comparison in this case.

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

Bits 3:0 **BKAWH[3:0]**: Break signal assignment to analog watchdog high threshold event

BKAWH[i] = 0: Break i signal is not assigned to an analog watchdog high threshold event

BKAWH[i] = 1: Break i signal is assigned to an analog watchdog high threshold event

26.8.10 DFSDM filter x analog watchdog low threshold register (DFSDM_FLTxAWLTR)

Address offset: 0x124 + 0x80 * 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
AWLT[23:8]															
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
AWLT[7:0]								Res.	Res.	Res.	Res.	BKAWL[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:8 **AWLT[23:0]**: Analog watchdog low threshold

These bits are written by software to define the low threshold for the analog watchdog.

Note: In case channel transceivers monitor (AWFSEL=1), only the higher 16 bits (AWLT[23:8]) define the 16-bit threshold as compared with the analog watchdog filter output (because data coming from the analog watchdog filter are up to a 16-bit resolution). Bits AWLT[7:0] are not taken into comparison in this case.

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

Bits 3:0 **BKAWL[3:0]**: Break signal assignment to analog watchdog low threshold event

BKAWL[i] = 0: Break i signal is not assigned to an analog watchdog low threshold event

BKAWL[i] = 1: Break i signal is assigned to an analog watchdog low threshold event

26.8.11 DFSDM filter x analog watchdog status register (DFSDM_FLTxAWSR)

Address offset: 0x128 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **AWHTF[3:0]**: Analog watchdog high threshold flag

AWHTF[y]=1 indicates a high threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWHTF[y] bit in the DFSDM_FLTxAWCFR register.

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

Bits 3:0 **AWLTF[3:0]**: Analog watchdog low threshold flag

AWLTF[y]=1 indicates a low threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWLTF[y] bit in the DFSDM_FLTxAWCFR register.

Note: All the bits of DFSDM_FLTxAWSR are automatically reset when DFEN=0.

26.8.12 DFSDM filter x analog watchdog clear flag register (DFSDM_FLTxAWCFR)

Address offset: 0x12C + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CLRAWHTF[3:0]				Res.	Res.	Res.	Res.	CLRAWLTF[3:0]			
				rc_w1	rc_w1	rc_w1	rc_w1					rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **CLRAWHTF[3:0]**: Clear the analog watchdog high threshold flag
CLRAWHTF[y]=0: Writing ‘0’ has no effect
CLRAWHTF[y]=1: Writing ‘1’ to position y clears the corresponding AWHTF[y] bit in the DFSDM_FLTxAWSR register

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

Bits 3:0 **CLRAWLTF[3:0]**: Clear the analog watchdog low threshold flag
CLRAWLTF[y]=0: Writing ‘0’ has no effect
CLRAWLTF[y]=1: Writing ‘1’ to position y clears the corresponding AWLTF[y] bit in the DFSDM_FLTxAWSR register

26.8.13 DFSDM filter x extremes detector maximum register (DFSDM_FLTxEXMAX)

Address offset: 0x130 + 0x80 * x, (x = 0 to 3)

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXMAX[23:8]															
rs_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	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
EXMAX[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	EXMAXCH[1:0]	
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r							r	r

Bits 31:8 **EXMAX[23:0]**: Extremes detector maximum value
These bits are set by hardware and indicate the highest value converted by DFSDM_FLTx. EXMAX[23:0] bits are reset to value (0x800000) by reading of this register.

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

Bits 1:0 **EXMAXCH[1:0]**: Extremes detector maximum data channel.
These bits contains information about the channel on which the data is stored into EXMAX[23:0]. Bits are cleared by reading of this register.

26.8.14 DFSDM filter x extremes detector minimum register (DFSDM_FLTxEXMIN)

Address offset: 0x134 + 0x80 * x, (x = 0 to 3)

Reset value: 0x7FFF FF00

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

Bits 31:8 **EXMIN[23:0]**: Extremes detector minimum value
These bits are set by hardware and indicate the lowest value converted by DFSDM_FLTx. EXMIN[23:0] bits are reset to value (0x7FFFFF) by reading of this register.

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

Bits 1:0 **EXMINCH[1:0]**: Extremes detector minimum data channel
These bits contain information about the channel on which the data is stored into EXMIN[23:0]. Bits are cleared by reading of this register.

26.8.15 DFSDM filter x conversion timer register (DFSDM_FLTxCNVTIMR)

Address offset: 0x138 + 0x80 * 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
CNVCNT[27:12]															
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
CNVCNT[11:0]												Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r				

Bits 31:4 **CNVCNT[27:0]**: 28-bit timer counting conversion time t = CNVCNT[27:0] / $f_{DFSDMCLK}$

The timer has an input clock from DFSDM clock (system clock $f_{DFSDMCLK}$). Conversion time measurement is started on each conversion start and stopped when conversion finishes (interval between first and last serial sample). Only in case of filter bypass (FOSR[9:0] = 0) is the conversion time measurement stopped and CNVCNT[27:0] = 0. The counted time is:

if FAST=0 (or first conversion in continuous mode if FAST=1):

$t = [F_{OSR} * (I_{OSR}\text{-}1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$..... for $Sinc^x$ filters

$t = [F_{OSR} * (I_{OSR}\text{-}1 + 4) + 2] / f_{CKIN}$..... for FastSinc filter

if FAST=1 in continuous mode (except first conversion):

$t = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case if F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator):

CNVCNT = 0 (counting is stopped, conversion time: $t = I_{OSR} / f_{CKIN}$)

where: f_{CKIN} is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input (from internal ADC or from CPU/DMA write)

Note: When conversion is interrupted (e.g. by disable/enable selected channel) the timer counts also this interruption time.

Bits 3:0 Reserved, must be kept at reset value.

26.8.16 DFSDM register map

The following table summarizes the DFSDM registers.

Table 205. DFSDM 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	**DFSDM_CH0CFGR1**	DFSDMEN	CKOUTSRC	Res.	Res.	Res.	Res.	Res.	Res.	CKOUTDIV[7:0]								DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[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
0x04	**DFSDM_CH0CFGR2**	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0																								0							
0x08	**DFSDM_CH0AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]	Res.	AWFOSR[4:0]						BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0
0x0C	**DFSDM_CH0WDATR**	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
0x10	**DFSDM_CH0DATINR**	INDAT1[15:0]																INDAT0[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
0x14	**DFSDM_CH0DLYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0
0x18 - 0x1C	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 205. DFSDM 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	**DFSDM_CH1CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x24	**DFSDM_CH1CFGR2**	OFFSET[23:0]																								DTRBS[4:0]					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	0	0	0	0	0			
0x28	**DFSDM_CH1AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0
0x2C	**DFSDM_CH1WDATR**	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	0
0x30	**DFSDM_CH1DATINR**	INDAT1[15:0]																INDAT0[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
0x34	**DFSDM_CH1DLYR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0
0x38 - 0x3C	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.
0x40	**DFSDM_CH2CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x44	**DFSDM_CH2CFGR2**	OFFSET[23:0]																								DTRBS[4:0]					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	0	0	0	0	0			
0x48	**DFSDM_CH2AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0
0x4C	**DFSDM_CH2WDATR**	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
0x50	**DFSDM_CH2DATINR**	INDAT1[15:0]																INDAT0[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
0x54	**DFSDM_CH2DLYR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0
0x58 - 0x5C	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 205. DFSDM 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	**DFSDM_ CH3CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x64	**DFSDM_ CH3CFGR2**	OFFSET[23:0]																								DTRBS[4:0]					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	0	0	0	0	0			
0x68	**DFSDM_ CH3AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x6C	**DFSDM_ CH3WDATR**	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
0x70	**DFSDM_ CH3DATINR**	INDAT1[15:0]																INDAT0[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
0x74	**DFSDM_ CH3DLYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0
0x78 - 0xFC	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	**DFSDM_ FLT0CR1**	Res.	AWFSEL	FAST	Res.	Res.	Res.	RCH[1:0]		Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]				Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN	
	reset value		0	0				0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x104	**DFSDM_ FLT0CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[3:0]				Res.	Res.	Res.	Res.	EXCH[3:0]				Res.	CKABIE	SCDIE	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value													0	0	0	0					0	0	0	0		0	0	0	0	0	0	0
0x108	**DFSDM_ FLT0ISR**	Res.	Res.	Res.	Res.	SCDF[3:0]				Res.	Res.	Res.	Res.	CKABF[3:0]				Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value					0	0	0	0					1	1	1	1		0	0									0	0	0	0	0
0x10C	**DFSDM_ FLT0ICR**	Res.	Res.	Res.	Res.	CLRSCDF [3:0]				Res.	Res.	Res.	Res.	CLRCKABF [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value					0	0	0	0					0	0	0	0													0	0		
0x110	**DFSDM_ FLT0JCHGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[3:0]			
	reset value																													0	0	0	1
0x114	**DFSDM_ FLT0FCR**	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[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

Table 205. DFSDM 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
0x118	**DFSDM_FLT0JDATAR**	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	JDATACH [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
0x11C	**DFSDM_FLT0RDATAR**	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	Res.	RDATACH[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
0x120	**DFSDM_FLT0AWHTR**	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[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
0x124	**DFSDM_FLT0AWLTR**	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[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
0x128	**DFSDM_FLT0AWSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[3:0]				Res.	Res.	Res.	Res.	AWLTF[3:0]			
	reset value																					0	0	0	0					0	0	0	0
0x12C	**DFSDM_FLT0AWCFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF [3:0]				Res.	Res.	Res.	Res.	CLRAWLTF [3:0]			
	reset value																					0	0	0	0					0	0	0	0
0x130	**DFSDM_FLT0EXMAX**	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	EXMAXCH[1:0]	
	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
0x134	**DFSDM_FLT0EXMIN**	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	EXMINCH[1:0]	
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1							0	0
0x138	**DFSDM_FLT0CNVTIMR**	CNVCNT[27:0]																												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	0	0	0	0				
0x13C - 0x17C	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.
0x180	**DFSDM_FLT1CR1**	Res.	AWFSEL	FAST	Res.	Res.	Res.	RCH[1:0]		Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0				0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x184	**DFSDM_FLT1CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[3:0]				Res.	Res.	Res.	Res.	EXCH[3:0]				Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value													0	0	0	0					0	0	0	0				0	0	0	0	0

Table 205. DFSDM 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
0x188	**DFSDM_ FLT1ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x18C	**DFSDM_ FLT1ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		
0x190	**DFSDM_ FLT1JCHGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[3:0]			
	reset value																													0	0	0	1
0x194	**DFSDM_ FLT1FCR**	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[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
0x198	**DFSDM_ FLT1JDATAR**	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	JDATACH[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
0x19C	**DFSDM_ FLT1RDATAR**	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	Res.	RDATACH[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
0x1A0	**DFSDM_ FLT1AWHTR**	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[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
0x1A4	**DFSDM_ FLT1AWLTR**	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[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
0x1A8	**DFSDM_ FLT1AWSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[3:0]				Res.	Res.	Res.	Res.	AWLTF[3:0]			
	reset value																					0	0	0	0					0	0	0	0
0x1AC	**DFSDM_ FLT1AWCFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF [3:0]				Res.	Res.	Res.	Res.	CLRAWLTF [3:0]			
	reset value																					0	0	0	0					0	0	0	0
0x1B0	**DFSDM_ FLT1EXMAX**	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	EXMAXCH[1:0]	
	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
0x1B4	**DFSDM_ FLT1EXMIN**	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	EXMINCH[1:0]	
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1							0	0

Table 205. DFSDM 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
0x1B8	**DFSDM_ FLT1CNVTIMR**	CNVCNT[27:0]																												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	0	0	0	0				
0x1BC - 0x1FC	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.
0x200	**DFDM_ FLT2CR1**	Res.	AWFSEL	FAST	Res.	Res.	Res.	RCH[1:0]		Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0				0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x204	**DFSDM_ FLT2CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[3:0]				Res.	Res.	Res.	Res.	EXCH[3:0]				Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value													0	0	0	0					0	0	0	0				0	0	0	0	0
0x208	**DFSDM_ FLT2ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x20C	**DFSDM_ FLT2ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		
0x210	**DFSDM_ FLT2JCHGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[3:0]			
	reset value																													0	0	0	1
0x214	**DFSDM_ FLT2FCR**	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[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
0x218	**DFSDM_ FLT2JDATAR**	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATACH[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
0x21C	**DFSDM_ FLT2RDATAR**	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	Res.	RDATACH[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
0x220	**DFSDM_ FLT2AWHTR**	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[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
0x224	**DFSDM_ FLT2AWLTR**	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[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
0x228	**DFSDM_ FLT2AWSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[3:0]				Res.	Res.	Res.	Res.	AWLTF[3:0]			
	reset value																					0	0	0	0					0	0	0	0

Table 205. DFSDM 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
0x22C	**DFSDM_ FLT2AWCFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF [3:0]				Res.	Res.	Res.	Res.	CLRAWLTF [3:0]			
	reset value																					0	0	0	0					0	0	0	0
0x230	**DFSDM_ FLT2EXMAX**	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	EXMAXCH[1:0]	
	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
0x234	**DFSDM_ FLT2EXMIN**	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	EXMINCH[1:0]	
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1							0	0
0x238	**DFSDM_ FLT2CNVTIMR**	CNVCNT[27:0]																												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	0	0	0	0				
0x23C - 0x27C	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.
0x280	**DFSDM_ FLT3CR1**	Res.	AWFSEL	FAST	Res.	Res.	Res.	RCH[1:0]		Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0				0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x284	**DFSDM_ FLT3CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[3:0]				Res.	Res.	Res.	Res.	EXCH[3:0]				Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value													0	0	0	0					0	0	0	0				0	0	0	0	0
0x288	**DFSDM_ FLT3ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x28C	**DFSDM_ FLT3ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		
0x290	**DFSDM_ FLT3JCHGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[3:0]			
	reset value																													0	0	0	1
0x294	**DFSDM_ FLT3FCR**	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[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
0x298	**DFSDM_ FLT3JDATAR**	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	JDATACH[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

Table 205. DFSDM 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
0x29C	**DFSDM_ FLT3RDATAR**	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	Res.	RDATACH[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
0x2A0	**DFSDM_ FLT3AWHTR**	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[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
0x2A4	**DFSDM_ FLT3AWLTR**	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[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
0x2A8	**DFSDM_ FLT3AWSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[3:0]				Res.	Res.	Res.	Res.	AWLTF[3:0]			
	reset value																					0	0	0	0					0	0	0	0
0x2AC	**DFSDM_ FLT3AWCFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF [3:0]				Res.	Res.	Res.	Res.	CLRAWLTF [3:0]			
	reset value																					0	0	0	0					0	0	0	0
0x2B0	**DFSDM_ FLT3EXMAX**	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	EXMAXCH[1:0]	
	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
0x2B4	**DFSDM_ FLT3EXMIN**	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	EXMINCH[1:0]	
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1							0	0
0x2B8	**DFSDM_ FLT3CNVTIMR**	CNVCNT[27:0]																												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	0	0	0	0				
0x2BC - 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 2.3 on page 87* for the register boundary addresses.

27 Touch sensing controller (TSC)

27.1 Introduction

The touch sensing controller provides a simple solution for adding capacitive sensing functionality to any application. Capacitive sensing technology is able to detect finger presence near an electrode that is protected from direct touch by a dielectric (for example glass, plastic). The capacitive variation introduced by the finger (or any conductive object) is measured using a proven implementation based on a surface charge transfer acquisition principle.

The touch sensing controller is fully supported by the STMTouch touch sensing firmware library, which is free to use and allows touch sensing functionality to be implemented reliably in the end application.

27.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 24 capacitive sensing channels
- Up to 8 capacitive sensing channels can be acquired in parallel offering a very good response time
- Spread spectrum feature to improve system robustness in noisy environments
- Full hardware management of the charge transfer acquisition sequence
- Programmable charge transfer frequency
- Programmable sampling capacitor I/O pin
- Programmable channel I/O pin
- Programmable max count value to avoid long acquisition when a channel is faulty
- Dedicated end of acquisition and max count error flags with interrupt capability
- One sampling capacitor for up to 3 capacitive sensing channels to reduce the system components
- Compatible with proximity, touchkey, linear and rotary touch sensor implementation
- Designed to operate with STMTouch touch sensing firmware library

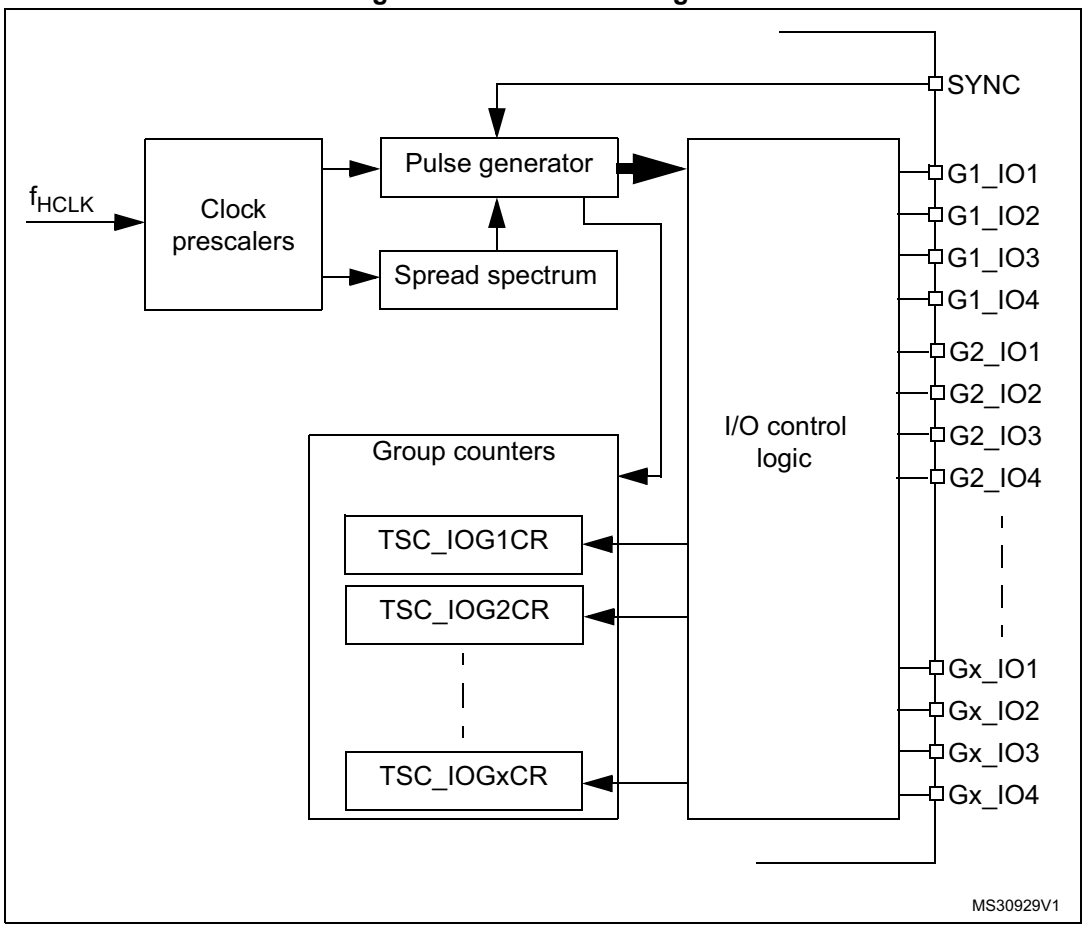
Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to IO availability.

27.3 TSC functional description

27.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in *Figure 186*.

Figure 186. TSC block diagram



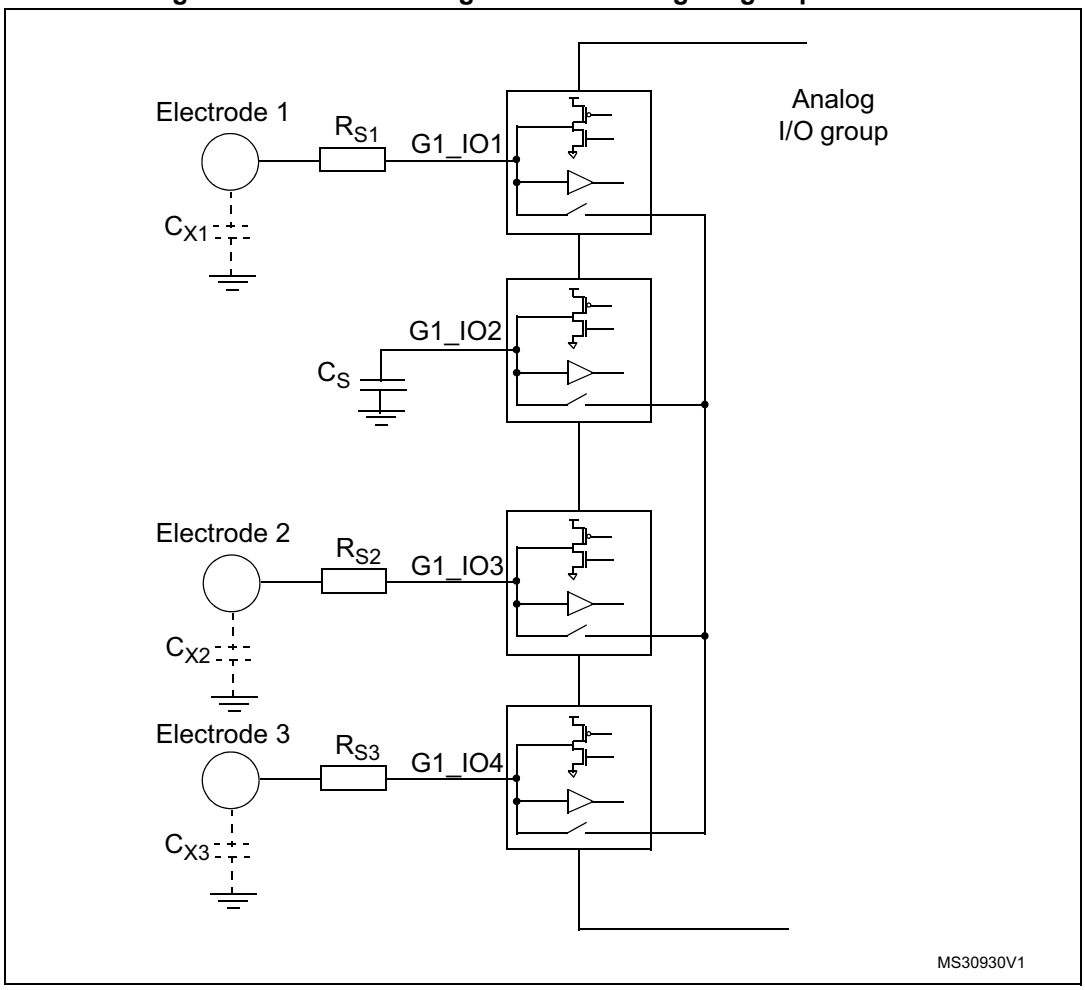
27.3.2 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum number of external components to operate with a single ended electrode type. This acquisition is designed around an analog I/O group composed of up to four GPIOs (see *Figure 187*). Several analog I/O groups are available to allow the acquisition of several capacitive sensing channels simultaneously and to support a larger number of capacitive sensing channels. Within a same analog I/O group, the acquisition of the capacitive sensing channels is sequential.

One of the GPIOs is dedicated to the sampling capacitor C_S. Only one sampling capacitor I/O per analog I/O group must be enabled at a time.

The remaining GPIOs are dedicated to the electrodes and are commonly called channels. For some specific needs (such as proximity detection), it is possible to simultaneously enable more than one channel per analog I/O group.

Figure 187. Surface charge transfer analog I/O group structure



Note: Gx_IOy where x is the analog I/O group number and y the GPIO number within the selected group.

The surface charge transfer acquisition principle consists of charging an electrode capacitance (C_X) and transferring a part of the accumulated charge into a sampling capacitor (C_S). This sequence is repeated until the voltage across C_S reaches a given threshold (V_{IH} in our case). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

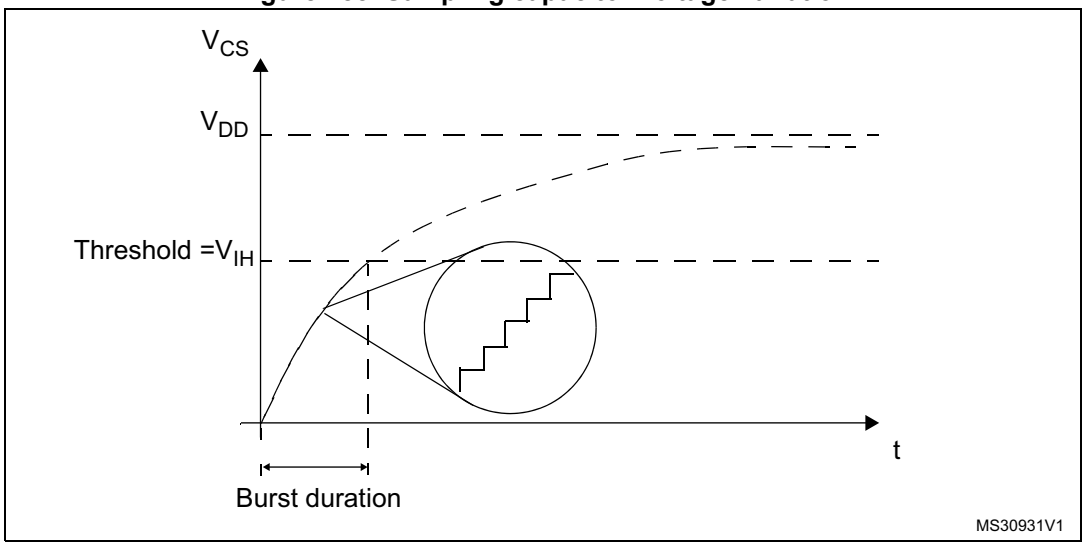
Table 206 details the charge transfer acquisition sequence of the capacitive sensing channel 1. States 3 to 7 are repeated until the voltage across C_S reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor R_S improves the ESD immunity of the solution.

Table 206. Acquisition sequence summary

State	G1_IO1 (channel)	G1_IO2 (sampling)	G1_IO3 (channel)	G1_IO4 (channel)	State description
#1	Input floating with analog switch closed	Output open-drain low with analog switch closed	Input floating with analog switch closed		Discharge all C_X and C_S
#2	Input floating				Dead time
#3	Output push-pull high	Input floating			Charge C_{X1}
#4	Input floating				Dead time
#5	Input floating with analog switch closed		Input floating		Charge transfer from C_{X1} to C_S
#6	Input floating				Dead time
#7	Input floating				Measure C_S voltage

The voltage variation over the time on the sampling capacitor C_S is detailed below:

Figure 188. Sampling capacitor voltage variation



27.3.3 Reset and clocks

The TSC clock source is the AHB clock (HCLK). Two programmable prescalers are used to generate the pulse generator and the spread spectrum internal clocks:

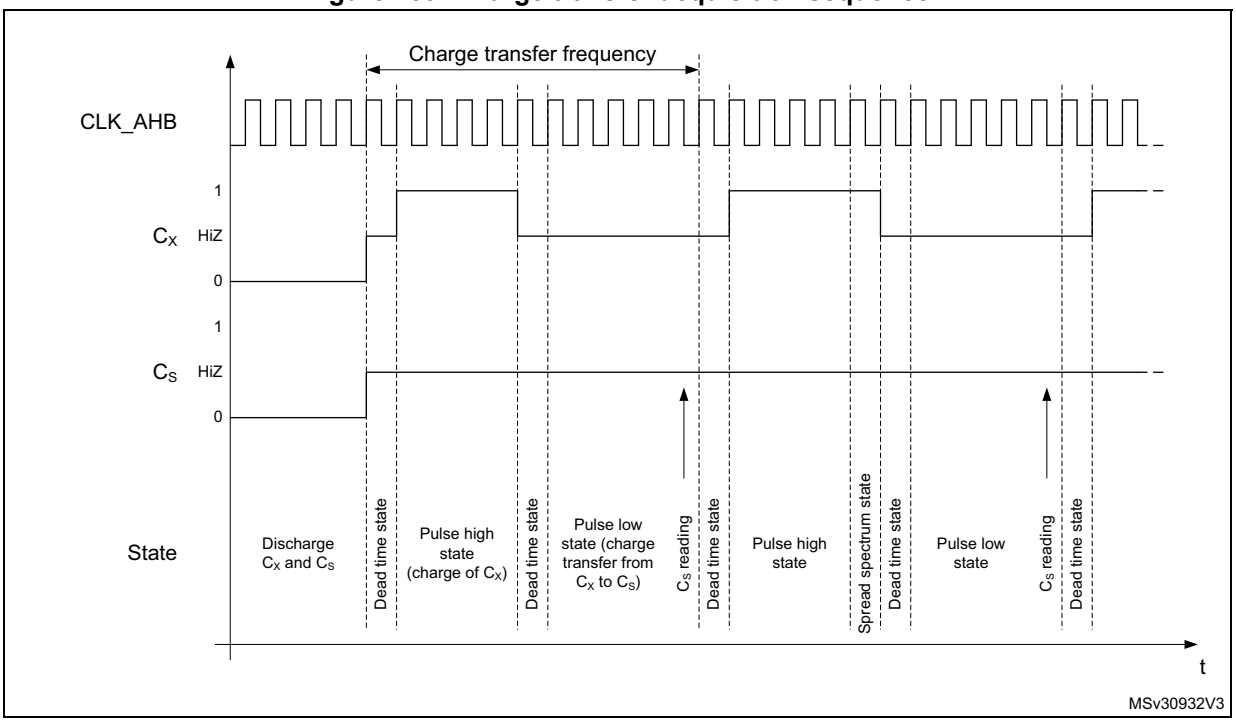
- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC_CR register

The Reset and Clock Controller (RCC) provides dedicated bits to enable the touch sensing controller clock and to reset this peripheral. For more information, refer to *Section 9: Reset and clock control (RCC)*.

27.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in *Figure 189*.

Figure 189. Charge transfer acquisition sequence



For higher flexibility, the charge transfer frequency is fully configurable. Both the pulse high state (charge of C_X) and the pulse low state (transfer of charge from C_X to C_S) duration can be defined using the CTPH[3:0] and CTPL[3:0] bits in the TSC_CR register. The standard range for the pulse high and low states duration is 500 ns to 2 µs. To ensure a correct measurement of the electrode capacitance, the pulse high state duration must be set to ensure that C_X is always fully charged.

A dead time where both the sampling capacitor I/O and the channel I/O are in input floating state is inserted between the pulse high and low states to ensure an optimum charge transfer acquisition sequence. This state duration is 1 periods of HCLK.

At the end of the pulse high state and if the spread spectrum feature is enabled, a variable number of periods of the SSCLK clock are added.

The reading of the sampling capacitor I/O, to determine if the voltage across C_S has reached the given threshold, is performed at the end of the pulse low state.

Note: The following TSC control register configurations are forbidden:

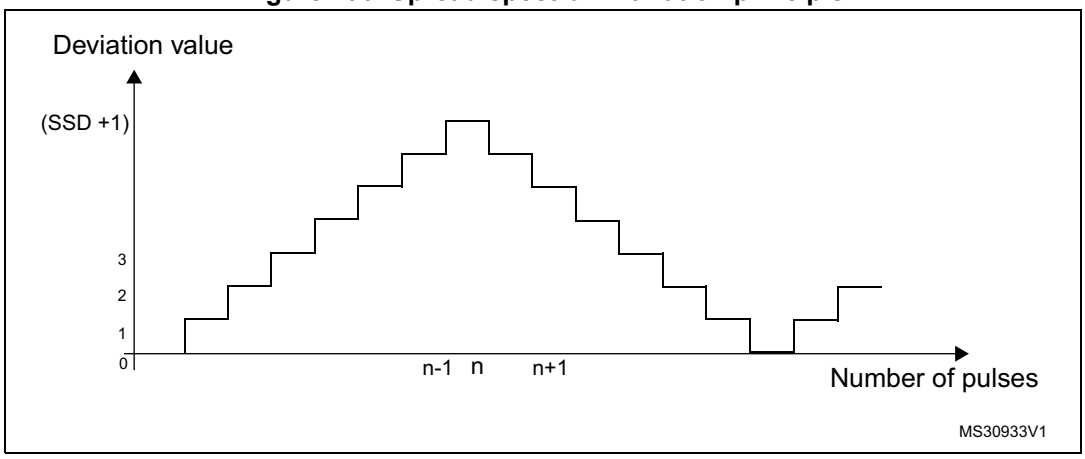
- *bits PGPSC are set to '000' and bits CTPL are set to '0000'*
- *bits PGPSC are set to '000' and bits CTPL are set to '0001'*
- *bits PGPSC are set to '001' and bits CTPL are set to '0000'*

27.3.5 Spread spectrum feature

The spread spectrum feature allows to generate a variation of the charge transfer frequency. This is done to improve the robustness of the charge transfer acquisition in noisy environments and also to reduce the induced emission. The maximum frequency variation is in the range of 10% to 50% of the nominal charge transfer period. For instance, for a nominal charge transfer frequency of 250 kHz (4 µs), the typical spread spectrum deviation is 10% (400 ns) which leads to a minimum charge transfer frequency of ~227 kHz.

In practice, the spread spectrum consists of adding a variable number of SSCLK periods to the pulse high state using the principle shown below:

Figure 190. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

Table 207. Spread spectrum deviation versus AHB clock frequency

f_{HCLK}	Spread spectrum step	Maximum spread spectrum deviation
24 MHz	41.6 ns	10666.6 ns
48 MHz	20.8 ns	5333.3 ns
80 MHz	12.5 ns	3205.1 ns
110 MHz	9.09 ns	2327.3 ns

The spread spectrum feature can be disabled/enabled using the SSE bit in the TSC_CR register. The frequency deviation is also configurable to accommodate the device HCLK clock frequency and the selected charge transfer frequency through the SSPSC and SSD[6:0] bits in the TSC_CR register.

27.3.6 Max count error

The max count error prevents long acquisition times resulting from a faulty capacitive sensing channel. It consists of specifying a maximum count value for the analog I/O group counters. This maximum count value is specified using the MCV[2:0] bits in the TSC_CR register. As soon as an acquisition group counter reaches this maximum value, the ongoing acquisition is stopped and the end of acquisition (EOAF bit) and max count error (MCEF bit) flags are both set. An interrupt can also be generated if the corresponding end of acquisition (EOAIE bit) or/and max count error (MCEIE bit) interrupt enable bits are set.

27.3.7 Sampling capacitor I/O and channel I/O mode selection

To allow the GPIOs to be controlled by the touch sensing controller, the corresponding alternate function must be enabled through the standard GPIO registers and the GPIOxAFR registers.

The GPIOs modes controlled by the TSC are defined using the TSC_IOSCR and TSC_IOCCR register.

When there is no ongoing acquisition, all the I/Os controlled by the touch sensing controller are in default state. While an acquisition is ongoing, only unused I/Os (neither defined as sampling capacitor I/O nor as channel I/O) are in default state. The IODEF bit in the TSC_CR register defines the configuration of the I/Os which are in default state. The table below summarizes the configuration of the I/O depending on its mode.

Table 208. I/O state depending on its mode and IODEF bit value

IODEF bit	Acquisition status	Unused I/O mode	Channel I/O mode	Sampling capacitor I/O mode
0 (output push-pull low)	No	Output push-pull low	Output push-pull low	Output push-pull low
0 (output push-pull low)	Ongoing	Output push-pull low	-	-
1 (input floating)	No	Input floating	Input floating	Input floating
1 (input floating)	Ongoing	Input floating	-	-

Unused I/O mode

An unused I/O corresponds to a GPIO controlled by the TSC peripheral but not defined as an electrode I/O nor as a sampling capacitor I/O.

Sampling capacitor I/O mode

To allow the control of the sampling capacitor I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output open drain mode and then the corresponding Gx_IOy bit in the TSC_IOSCR register must be set.

Only one sampling capacitor per analog I/O group must be enabled at a time.

Channel I/O mode

To allow the control of the channel I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output push-pull mode and the corresponding Gx_IOy bit in the TSC_IOCCR register must be set.

For proximity detection where a higher equivalent electrode surface is required or to speed-up the acquisition process, it is possible to enable and simultaneously acquire several channels belonging to the same analog I/O group.

Note: During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR or TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

27.3.8 Acquisition mode

The touch sensing controller offers two acquisition modes:

- Normal acquisition mode: the acquisition starts as soon as the START bit in the TSC_CR register is set.
- Synchronized acquisition mode: the acquisition is enabled by setting the START bit in the TSC_CR register but only starts upon the detection of a falling edge or a rising edge and high level on the SYNC input pin. This mode is useful for synchronizing the capacitive sensing channels acquisition with an external signal without additional CPU load.

The GxE bits in the TSC_IOGCSR registers specify which analog I/O groups are enabled (corresponding counter is counting). The C_S voltage of a disabled analog I/O group is not monitored and this group does not participate in the triggering of the end of acquisition flag. However, if the disabled analog I/O group contains some channels, they are pulsed.

When the C_S voltage of an enabled analog I/O group reaches the given threshold, the corresponding GxS bit of the TSC_IOGCSR register is set. When the acquisition of all enabled analog I/O groups is complete (all GxS bits of all enabled analog I/O groups are set), the EOAF flag in the TSC_ISR register is set. An interrupt request is generated if the EOAIE bit in the TSC_IER register is set.

In the case that a max count error is detected, the ongoing acquisition is stopped and both the EOAF and MCEF flags in the TSC_ISR register are set. Interrupt requests can be generated for both events if the corresponding bits (EOAIE and MCEIE bits of the TSCIER register) are set. Note that when the max count error is detected the remaining GxS bits in the enabled analog I/O groups are not set.

To clear the interrupt flags, the corresponding EOAIC and MCEIC bits in the TSC_ICR register must be set.

The analog I/O group counters are cleared when a new acquisition is started. They are updated with the number of charge transfer cycles generated on the corresponding channel(s) upon the completion of the acquisition.

27.3.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller also allows to take the control of the Schmitt trigger hysteresis and analog switch of each Gx_IOy. This control is available whatever the I/O control mode is (controlled by standard GPIO registers or other peripherals) assuming that the touch sensing controller is enabled. This may be useful to perform a different acquisition sequence or for other purposes.

In order to improve the system immunity, the Schmitt trigger hysteresis of the GPIOs controlled by the TSC must be disabled by resetting the corresponding Gx_IOy bit in the TSC_IOHCR register.

27.4 TSC low-power modes

Table 209. Effect of low-power modes on TSC

Mode	Description
Sleep	No effect. Peripheral interrupts cause the device to exit Sleep mode.
Low power run	No effect.
Low power sleep	No effect. Peripheral interrupts cause the device to exit Low-power sleep mode.
Stop 0 / Stop 1	Peripheral registers content is kept.
Stop 2	
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

27.5 TSC interrupts

Table 210. Interrupt control bits

Interrupt event	Enable control bit	Event flag	Clear flag bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
End of acquisition	EOAIE	EOAIF	EOAIC	Yes	No	No
Max count error	MCEIE	MCEIF	MCEIC	Yes	No	No

27.6 TSC registers

Refer to *Section 1.2 on page 76* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

27.6.1 TSC control register (TSC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTPH[3:0]				CTPL[3:0]				SSD[6:0]							SSE
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
SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV[2:0]			IODEF	SYNC POL	AM	START	TSCE
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **CTPH[3:0]**: Charge transfer pulse high

These bits are set and cleared by software. They define the duration of the high state of the charge transfer pulse (charge of C_X).

0000: 1x t_{PGCLK}
0001: 2x t_{PGCLK}
...
1111: 16x t_{PGCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Bits 27:24 **CTPL[3:0]**: Charge transfer pulse low

These bits are set and cleared by software. They define the duration of the low state of the charge transfer pulse (transfer of charge from C_X to C_S).

0000: 1x t_{PGCLK}
0001: 2x t_{PGCLK}
...
1111: 16x t_{PGCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Note: Some configurations are forbidden. Refer to the Section 27.3.4: Charge transfer acquisition sequence for details.

Bits 23:17 **SSD[6:0]**: Spread spectrum deviation

These bits are set and cleared by software. They define the spread spectrum deviation which consists in adding a variable number of periods of the SSCLK clock to the charge transfer pulse high state.

0000000: 1x t_{SSCLK}
0000001: 2x t_{SSCLK}
...
1111111: 128x t_{SSCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Bit 16 **SSE**: Spread spectrum enable

This bit is set and cleared by software to enable/disable the spread spectrum feature.

0: Spread spectrum disabled

1: Spread spectrum enabled

Note: This bit must not be modified when an acquisition is ongoing.

Bit 15 **SSPSC**: Spread spectrum prescaler

This bit is set and cleared by software. It selects the AHB clock divider used to generate the spread spectrum clock (SSCLK).

0: f_{HCLK}

1: f_{HCLK} /2

Note: This bit must not be modified when an acquisition is ongoing.

Bits 14:12 **PGPSC[2:0]**: Pulse generator prescaler

These bits are set and cleared by software.They select the AHB clock divider used to generate the pulse generator clock (PGCLK).

000: f_{HCLK}

001: f_{HCLK} /2

010: f_{HCLK} /4

011: f_{HCLK} /8

100: f_{HCLK} /16

101: f_{HCLK} /32

110: f_{HCLK} /64

111: f_{HCLK} /128

Note: These bits must not be modified when an acquisition is ongoing.

Note: Some configurations are forbidden. Refer to the Section 27.3.4: Charge transfer acquisition sequence for details.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **MCV[2:0]**: Max count value

These bits are set and cleared by software. They define the maximum number of charge transfer pulses that can be generated before a max count error is generated.

000: 255

001: 511

010: 1023

011: 2047

100: 4095

101: 8191

110: 16383

111: reserved

Note: These bits must not be modified when an acquisition is ongoing.

Bit 4 **IODEF**: I/O Default mode

This bit is set and cleared by software. It defines the configuration of all the TSC I/Os when there is no ongoing acquisition. When there is an ongoing acquisition, it defines the configuration of all unused I/Os (not defined as sampling capacitor I/O or as channel I/O).

0: I/Os are forced to output push-pull low

1: I/Os are in input floating

Note: This bit must not be modified when an acquisition is ongoing.

Bit 3 **SYNCPOL**: Synchronization pin polarity

This bit is set and cleared by software to select the polarity of the synchronization input pin.

0: Falling edge only

1: Rising edge and high level

Bit 2 **AM**: Acquisition mode

This bit is set and cleared by software to select the acquisition mode.

0: Normal acquisition mode (acquisition starts as soon as START bit is set)

1: Synchronized acquisition mode (acquisition starts if START bit is set and when the selected signal is detected on the SYNC input pin)

Note: This bit must not be modified when an acquisition is ongoing.

Bit 1 **START**: Start a new acquisition

This bit is set by software to start a new acquisition. It is cleared by hardware as soon as the acquisition is complete or by software to cancel the ongoing acquisition.

0: Acquisition not started

1: Start a new acquisition

Bit 0 **TSCE**: Touch sensing controller enable

This bit is set and cleared by software to enable/disable the touch sensing controller.

0: Touch sensing controller disabled

1: Touch sensing controller enabled

Note: When the touch sensing controller is disabled, TSC registers settings have no effect.

27.6.2 TSC interrupt enable register (TSC_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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIE	EOAIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIE**: Max count error interrupt enable

This bit is set and cleared by software to enable/disable the max count error interrupt.

0: Max count error interrupt disabled

1: Max count error interrupt enabled

Bit 0 **EOAIE**: End of acquisition interrupt enable

This bit is set and cleared by software to enable/disable the end of acquisition interrupt.

0: End of acquisition interrupt disabled

1: End of acquisition interrupt enabled

27.6.3 TSC interrupt clear register (TSC_ICR)

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.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIC	EOAIC
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIC**: Max count error interrupt clear

This bit is set by software to clear the max count error flag and it is cleared by hardware when the flag is reset. Writing a ‘0’ has no effect.

0: No effect

1: Clears the corresponding MCEF of the TSC_ISR register

Bit 0 **EOAIC**: End of acquisition interrupt clear

This bit is set by software to clear the end of acquisition flag and it is cleared by hardware when the flag is reset. Writing a ‘0’ has no effect.

0: No effect

1: Clears the corresponding EOAF of the TSC_ISR register

27.6.4 TSC interrupt status register (TSC_ISR)

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.	Res.	Res.	MCEF	EOAF
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEF**: Max count error flag

This bit is set by hardware as soon as an analog I/O group counter reaches the max count value specified. It is cleared by software writing 1 to the bit MCEIC of the TSC_ICR register.

0: No max count error (MCE) detected

1: Max count error (MCE) detected

Bit 0 **EOAF**: End of acquisition flag

This bit is set by hardware when the acquisition of all enabled group is complete (all GxS bits of all enabled analog I/O groups are set or when a max count error is detected). It is cleared by software writing 1 to the bit EOAIC of the TSC_ICR register.

0: Acquisition is ongoing or not started

1: Acquisition is complete

27.6.5 TSC I/O hysteresis control register (TSC_IOHCR)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx_IOy**: Gx_IOy Schmitt trigger hysteresis mode

These bits are set and cleared by software to enable/disable the Gx_IOy Schmitt trigger hysteresis.

0: Gx_IOy Schmitt trigger hysteresis disabled

1: Gx_IOy Schmitt trigger hysteresis enabled

Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).

27.6.6 TSC I/O analog switch control register (TSC_IOASCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx_IOy**: Gx_IOy analog switch enable

These bits are set and cleared by software to enable/disable the Gx_IOy analog switch.

0: Gx_IOy analog switch disabled (opened)

1: Gx_IOy analog switch enabled (closed)

Note: These bits control the I/O analog switch whatever the I/O control mode is (even if controlled by standard GPIO registers).

27.6.7 TSC I/O sampling control register (TSC_IOSCR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx_IOy**: Gx_IOy sampling mode

These bits are set and cleared by software to configure the Gx_IOy as a sampling capacitor I/O. Only one I/O per analog I/O group must be defined as sampling capacitor.

0: Gx_IOy unused

1: Gx_IOy used as sampling capacitor

Note: These bits must not be modified when an acquisition is ongoing.

During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

27.6.8 TSC I/O channel control register (TSC_IOCCR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx_IOy**: Gx_IOy channel mode

These bits are set and cleared by software to configure the Gx_IOy as a channel I/O.
0: Gx_IOy unused
1: Gx_IOy used as channel

Note: These bits must not be modified when an acquisition is ongoing.
During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

27.6.9 TSC I/O group control status register (TSC_IOGCSR)

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.	G8S	G7S	G6S	G5S	G4S	G3S	G2S	G1S
								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.	G8E	G7E	G6E	G5E	G4E	G3E	G2E	G1E
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **GxS**: Analog I/O group x status

These bits are set by hardware when the acquisition on the corresponding enabled analog I/O group x is complete. They are cleared by hardware when a new acquisition is started.
0: Acquisition on analog I/O group x is ongoing or not started
1: Acquisition on analog I/O group x is complete

Note: When a max count error is detected the remaining GxS bits of the enabled analog I/O groups are not set.

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

Bits 7:0 **GxE**: Analog I/O group x enable

These bits are set and cleared by software to enable/disable the acquisition (counter is counting) on the corresponding analog I/O group x.
0: Acquisition on analog I/O group x disabled
1: Acquisition on analog I/O group x enabled

27.6.10 TSC I/O group x counter register (TSC_IOGxCR)

x represents the analog I/O group number.

Address offset: 0x30 + 0x04 * x, (x = 1..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
Res.	Res.	CNT[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 **CNT[13:0]**: Counter value

These bits represent the number of charge transfer cycles generated on the analog I/O group x to complete its acquisition (voltage across C_S has reached the threshold).

27.6.11 TSC register map

Table 211. TSC 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
0x0000	**TSC_CR**	CTPH[3:0]				CTPL[3:0]				SSD[6:0]							SSE	SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV [2:0]			IODEF	SYNCPOL	AM	START	TSCE
	Reset value	0	0	0	0	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	**TSC_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.	Res.	Res.	Res.	Res.	Res.	MCEIE	EOAIE
	Reset value																															0	0
0x0008	**TSC_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.	Res.	Res.	MCEIC	EOAIC
	Reset value																															0	0
0x000C	**TSC_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.	Res.	Res.	Res.	Res.	Res.	MCEF	EOAF
	Reset value																															0	0
0x0010	**TSC_IOHCR**	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	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
0x0014	Reserved																																
0x0018	**TSC_IOASCR**	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value	0	0	0	0	0	0	0	0	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	Reserved																																
0x0020	**TSC_IOSCR**	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value	0	0	0	0	0	0	0	0	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	Reserved																																
0x0028	**TSC_IOCCR**	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value	0	0	0	0	0	0	0	0	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	Reserved																																
0x0030	**TSC_IOGCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8S	G7S	G6S	G5S	G4S	G3S	G2S	G1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8E	G7E	G6E	G5E	G4E	G3E	G2E	G1E
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x0034	**TSC_IOG1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	**TSC_IOG2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 211. TSC 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
0x003C	**TSC_IOG3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	**TSC_IOG4CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	**TSC_IOG5CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	**TSC_IOG6CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	**TSC_IOG7CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	**TSC_IOG8CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3 on page 87* for the register boundary addresses.

28 True random number generator (RNG)

28.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is a NIST SP 800-90B compliant entropy source that can be used to construct a non-deterministic random bit generator (NDRBG).

The RNG true random number generator has been pre-certified NIST SP800-90B. It has also been tested using German BSI statistical tests of AIS-31 (T0 to T8).

28.2 RNG main features

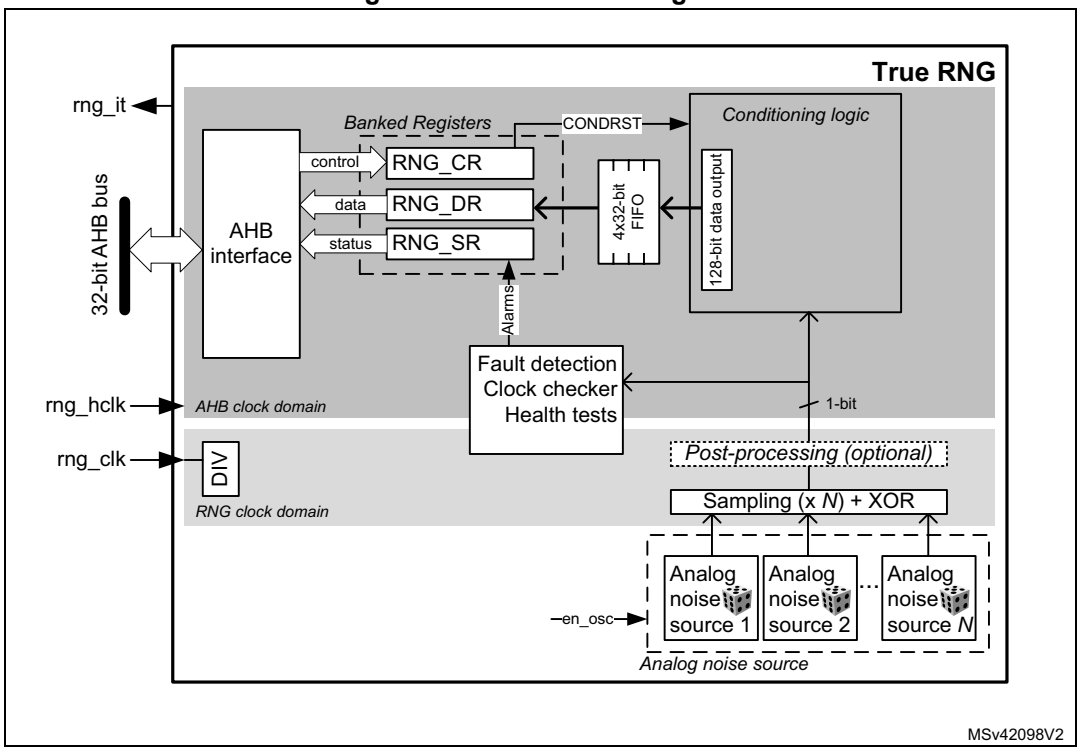
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as entropy source to construct a non-deterministic random bit generator (NDRBG).
- It produces four 32-bit random samples every 412 AHB clock cycles if f_{AHB} < 77 MHz (256 RNG clock cycles otherwise).
- It embeds start-up and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- It has an AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

28.3 RNG functional description

28.3.1 RNG block diagram

Figure 191 shows the RNG block diagram.

Figure 191. RNG block diagram



28.3.2 RNG internal signals

Table 212 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 212. RNG internal input/output signals

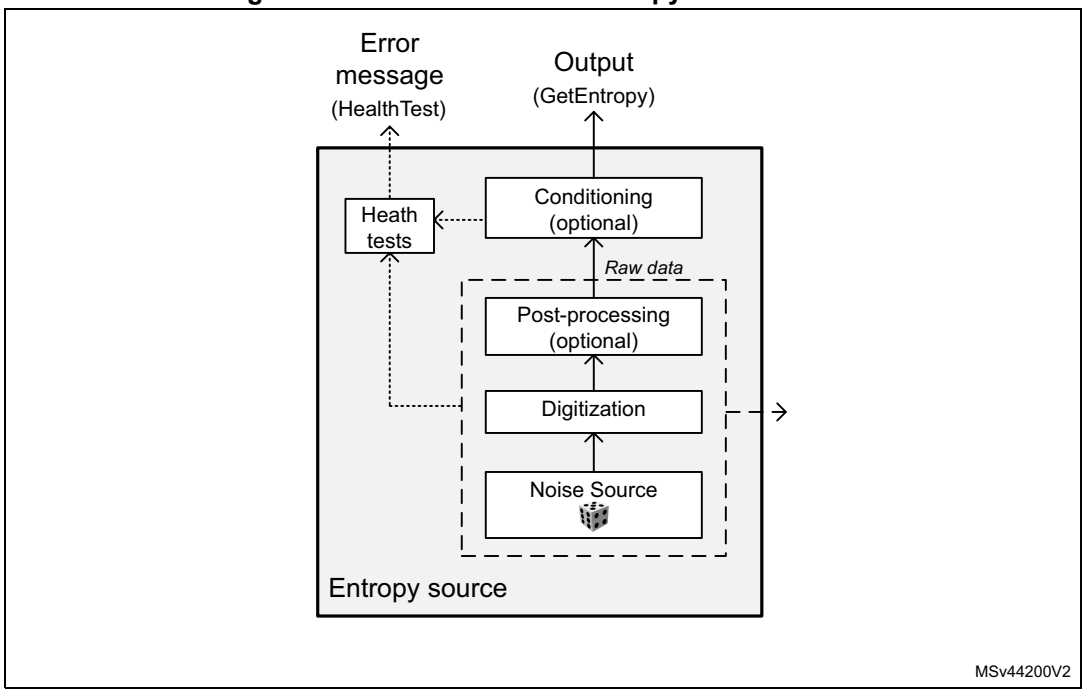
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk

28.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary RNG integrates all the required NIST components depicted on *Figure 192*. Those components are an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

Figure 192. NIST SP800-90B entropy source model



The components pictured above are detailed hereafter:

Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x6), each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 28.3.8: RNG low-power usage*.
- The XORing of the 6 noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (**rng_clk** with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (**rng_hclk**), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

Note: In Section 28.6: RNG entropy source validation recommended RNG clock frequencies and associated divider value are given.

Post processing

In NIST configuration no post-processing is applied to sampled noise source. In non-NIST configuration B (as defined in *Section 28.6.2*) a normalization debiasing is applied, i.e. half of the bits are taken from the sampled noise source, half of the bits are taken from inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit). The NIST SP800-90B target is full entropy on the output (i.e. 128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in *Section 28.5: RNG processing time*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 28.5: RNG processing time*.

Whenever a random number is available through the RNG_DR register the DRDY flag transitions from “0” to “1”. This flag remains high until output buffer becomes empty after reading four words from the RNG_DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from “0” to “1”. Interrupt is then cleared automatically by the RNG as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (in *Section 28.6.2*).

1. *Start-up health tests, performed after reset and before the first use of the RNG as entropy source*
   - Adaptive proportion test running on one 1024 bit windows: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 691 times.
   - Known-answer tests, to verify the conditioning stage.
   - Repetition count test, flagging an error when the noise source has provided more than 40 consecutive bits at a constant value (“0” or “1”)
2. Continuous health tests, running indefinitely on the outputs of the noise source
   - Repetition count test, similar to the one running in start-up tests
   - Adaptive proportion test running on 1024 consecutive samples, like during start-up health tests.
3. Vendor specific continuous tests
   - Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrence of two bits patterns (“01” or “10”).
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than AHB clock cycle divided by 32.
4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and re-running the startup tests (see software reset sequence in *Section 28.3.4: RNG initialization*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 28.3.7: Error management*.

Note: An interrupt can be generated when an error is detected.

Above health test thresholds are modified by changing value in RNG_HTCR register. See Section 28.6: RNG entropy source validation for details.

28.3.4 RNG initialization

The RNG simplified state machine is pictured on *Figure 193*

After enabling the RNG (RNGEN=1 in RNG_CR) the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 cycles of RNG clock cycles (before divider) before starting to sample analog output and filling 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering start-up behavior test on the raw data samples and known-answer tests.
3. When start-up health tests are completed. During this time three 128-bit noise source samples are used.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 28.5: RNG processing time*.

Figure 193. RNG initialization overview

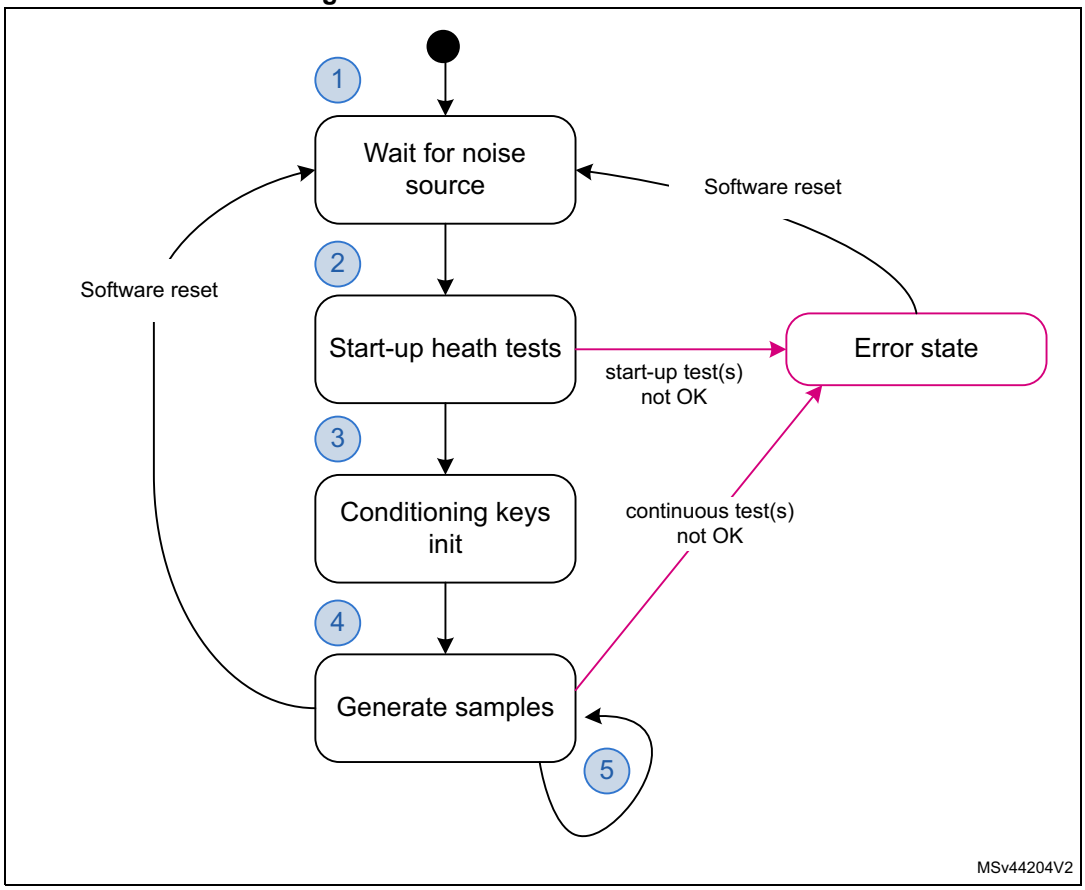


Figure 193 also highlights a possible software reset sequence, implemented by:

a) Writing bits RNGEN=0 and CONDRST=1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.

b) Then writing RNGEN=1 and CONDRST=0 in the RNG_CR register.

c) Wait for random number to be ready, after initialization completes

Note: When RNG peripheral is reset through RCC (hardware reset) the RNG configuration for optimal randomness is lost in RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

28.3.5 RNG operation

Normal operations

To run the RNG using interrupts the following steps are recommended:

1. Consult the *Section 28.6: RNG entropy source validation on page 949* and verify if a specific RNG configuration is required for your application.
   - If it is the case, write in the RNG_CR register the bit CONDRST="1" together with the correct RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST="0", the interrupt enable bit IE="1" and the RNG enable bit RNGEN="1".
   - If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE="1" and the RNG enable bit RNGEN="1".
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits 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 28.3.7* must be used.

To run the RNG in polling mode following steps are recommended:

1. Consult the *Section 28.6: RNG entropy source validation on page 949* and verify if a specific RNG configuration is required for your application.
   - If it is the case write in the RNG_CR register the bit CONDRST="1" together with the correction RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST="0" and the RNG enable bit RNGEN="1".
   - If it is not the case only enable the RNG by setting the RNGEN bit to "1" in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits 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 28.3.7* must be used.

Note: When data is not ready (DRDY="0") RNG_DR returns zero.
It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required it is possible to select a faster RNG configuration by using the RNG configuration "B", described in *Section 28.6: RNG entropy source validation*. The gain in random number generation speed is summarized in *Section 28.5: RNG processing time*.

Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 28.3.8: RNG low-power usage*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 28.3.3: Random number generation*.

28.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 28.6: RNG entropy source validation*.

*Note: When the CED bit in the RNG_CR register is set to "0", the RNG clock frequency before internal divider **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 28.3.1: RNG block diagram* for details (AHB and RNG clock domains).

28.3.7 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 28.3.6: 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.

The following sequence must be used to fully recover from a seed error:

1. Software reset by writing CONDRST at 1 and at 0 (see bitfield description for details).
2. wait for CONDRST to be cleared in the RNG_CR register, then confirm that SEIS is cleared in the RNG_SR register.
3. wait for SECS to be cleared by RNG. The random number generation is now back to normal.

28.3.8 RNG low-power usage

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to “1” by setting the RNGEN bit to “0” in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN=’0’ following features are available to software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode) and a complete conditioning round has been done. Four new random words are then available only if the expected number of conditioning round has been reached (two if NISTC=0). The overall time can be found in *Section 28.5: RNG processing time on page 949*.

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”, unless the application resets the conditioning logic using CONDRST bit in the RNG_CR register.

28.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 28.3.7: Error management*
- Clock error, see *Section 28.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 213*.

Table 213. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write 0 to SEIS or write CONDRST to 1 then to 0
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

28.5 RNG processing time

In NIST compliant configuration, the time between two sets of four 32-bit data is either:

- 412 AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 256 RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

With $f_{threshold} = 1.6 \times f_{AHB}$, e.g. 77 MHz if f_{RNG}= 48 MHz.

Note: When CLKDIV is different from zero, f_{RNG} must take into account the internal divider ratio.

28.6 RNG entropy source validation

28.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite. The results can be provided on demand or the customer can reproduce the tests.

28.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk= 48 MHz
- RNG configurations described in *Table 214*. Note that only configuration A can be certified NIST SP800-90B.

Table 214. RNG configurations

RNG Config	RNG_CR bits						Nb loop (N)	RNG_HTCR register(1)
	NISTC bit	RNG_CONFIG1 [5:0]	CLKDIV [3:0]	RNG_CONFIG2 [2:0]	RNG_CONFIG3 [3:0]	CED bit		
A	0	0x0F	0x0	0x0	0xD	0	2	0x0000 A2B3
B	1	0x18	0x0	0x0	0x0	0	2	

1. When writing this register magic number 0x17590ABC must be written immediately before the indicated value

28.6.3 Data collection

In order to run statistical tests it is required to collect samples from the entropy source at raw data level as well as at the output of the entropy source. For details on data collection and the running of statistical test suites refer to “STM32 microcontrollers random number generation validation using NIST statistical test suite” application note (AN4230) available from *www.st.com*.

Contact STMicroelectronics if above samples need to be retrieved for your product.

28.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

28.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	Res.	Res.	RNG_CONFIG1[5:0]						CLKDIV[3:0]			
rs	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw			rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG Config Lock

0: Writes to the RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

This bitfield is set once: if this bit is set it can only be reset to 0 if RNG is reset.

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_CR and RNG_NSCR are not changed by CONDRST.

This bit must be set to 1 in the same access that set any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

When CONDRST is set to 0 by software its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:20 **RNG_CONFIG1[5:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 28.6: RNG entropy source validation*.

Writing any bit of RNG_CONFIG1 is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK=1.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN=0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock, e.g. an incoming 48MHz RNG clock becomes a 1.5 kHz internal RNG clock.

Writing these bits is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK=1.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Refer to RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: Non NIST compliant

0: Hardware default values for NIST compliant RNG. In this configuration per 128-bit output two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 28.6: RNG entropy source validation* for proposed configuration.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK=1.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.
If NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection is enable

1: Clock error detection is disable

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, i.e. to enable or disable CED the RNG must be disabled.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK=1.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt Enable

0: RNG Interrupt is disabled

1: RNG Interrupt is enabled. An interrupt is pending as soon as DRDY='1', SEIS='1' or CEIS=1 in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.

1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.

28.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence has been detected. See **SECS** bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct (fRNGCLK> fHCLK/32)

1: The RNG clock before internal divider has been detected too slow (fRNGCLK< fHCLK/32)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

- Run-time repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0” or 1”, or more than 32 consecutive occurrence of two bits patterns 01” or 10”)
- Start-up or continuous adaptive proportion test on noise source failed.
- Start-up post-processing/conditioning sanity check failed.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct (fRNGCLK> fHCLK/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow (fRNGCLK< fHCLK/32).

Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data Ready

0: The RNG_DR register is not yet valid, no random data is available.

1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

Note: The DRDY bit can rise when the peripheral is disabled (RNGEN=0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY=1.

28.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when DRDY=1 and value is not 0x0, even if RNGEN=0.

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

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data which are valid when DRDY=1. When DRDY=0 RNDATA value is zero. 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).

28.7.4 RNG health test control register (RNG_HTCR)

Address offset: 0x010

Reset value: 0x000C AA74

Writing in RNG_HTCR is taken into account only if CONDRST bit is set, and CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK=1.

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

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests. See *Section 28.6: RNG entropy source validation* for the recommended value.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

When reading or writing this register magic number; 0x17590ABC must be written immediately before to RNG_HTCR register.

28.7.5 RNG register map

Table 215 gives the RNG register map and reset values.

Table 215. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	Res.	Res.	RNG_CONFIG1[5:0]						CLKDIV[3:0]				RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0					0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
	Reset value																										0	0			0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	RNG_HTCR	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	1	1	1	0	1	0	0

Refer to *Section 2.3* for the register boundary addresses.

29 AES hardware accelerator (AES)

29.1 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).

29.2 AES main features

- Compliance with NIST *"Advanced encryption standard (AES)*, *FIPS publication 197*" from November 2001
- 128-bit data block processing
- Support for cipher key lengths of 128-bit and 256-bit
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block of data with, respectively, 128-bit or 256-bit key
- Integrated round key scheduler to compute the last round key for ECB/CBC decryption
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only
- 256-bit write-only register for storing the cryptographic key (eight 32-bit registers)
- 128-bit register for storing initialization vector (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control with support of single-transfer direct memory access (DMA) using two channels (one for incoming data, one for processed data)
- Data-swapping logic to support 1-, 8-, 16- or 32-bit data
- Possibility for software to suspend a message if AES needs to process another message with a higher priority, then resume the original message

29.3 AES implementation

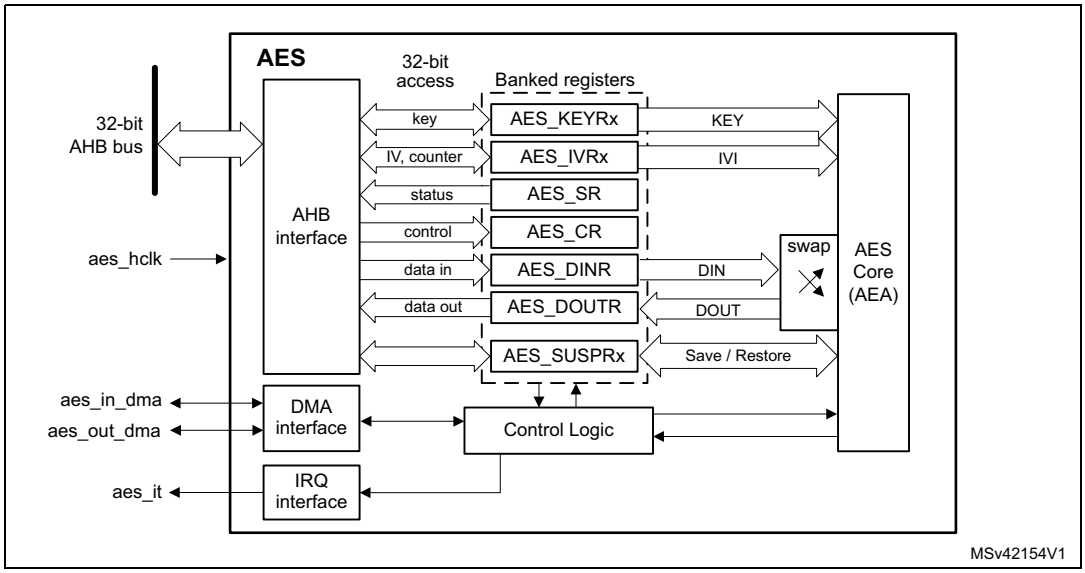
The devices have one AES peripheral.

29.4 AES functional description

29.4.1 AES block diagram

Figure 194 shows the block diagram of AES.

Figure 194. AES block diagram



29.4.2 AES internal signals

Table 216 describes the user relevant internal signals interfacing the AES peripheral.

Table 216. 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

29.4.3 AES cryptographic core

Overview

The AES cryptographic core consists of the following components:

- AES core algorithm (AEA)
- multiplier over a binary Galois field (GF2mul)
- key input
- initialization vector (IV) input
- chaining algorithm logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks (four words) with 128-bit or 256-bit key length. Depending on the chaining mode, the AES requires zero or one 128-bit initialization vector IV.

The AES features the following modes of operation:

- **Mode 1:**

  Plaintext encryption using a key stored in the AES_KEYRx registers
- **Mode 2:**

  ECB or CBC decryption key preparation. It must be used prior to selecting Mode 3 with ECB or CBC chaining modes. The key prepared for decryption is stored automatically in the AES_KEYRx registers. Now the AES peripheral is ready to switch to Mode 3 for executing data decryption.
- **Mode 3:**

  Ciphertext decryption using a key stored in the AES_KEYRx registers. When ECB and CBC chaining modes are selected, the key must be prepared beforehand, through Mode 2.

Note: Mode 2 is only used when performing ECB and CBC decryption.

The operating mode is selected by programming the MODE[1:0] bitfield of the AES_CR register. It may be done only when the AES peripheral is disabled.

Typical data processing

Typical usage of the AES is described in *Section 29.4.4: AES procedure to perform a cipher operation on page 963*.

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)

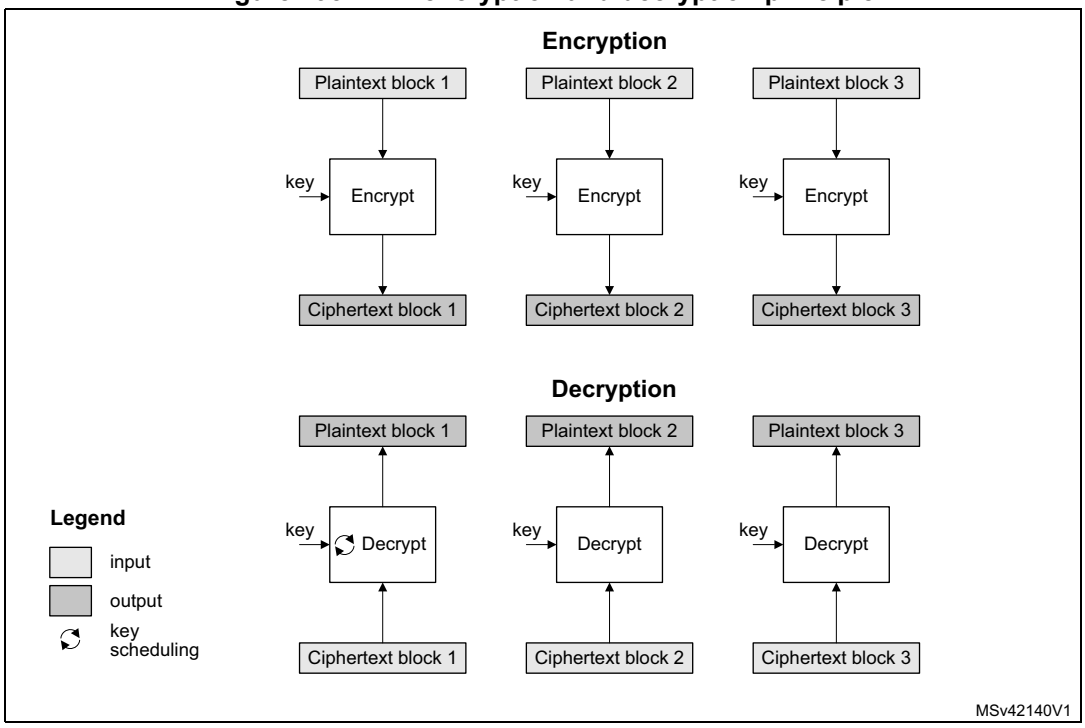
Note: The chaining mode may be changed only when AES is disabled (bit EN of the AES_CR register cleared).

Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from *Section 29.4.8: AES basic chaining modes (ECB, CBC)*.

Electronic codebook (ECB) mode

Figure 195. ECB encryption and decryption principle

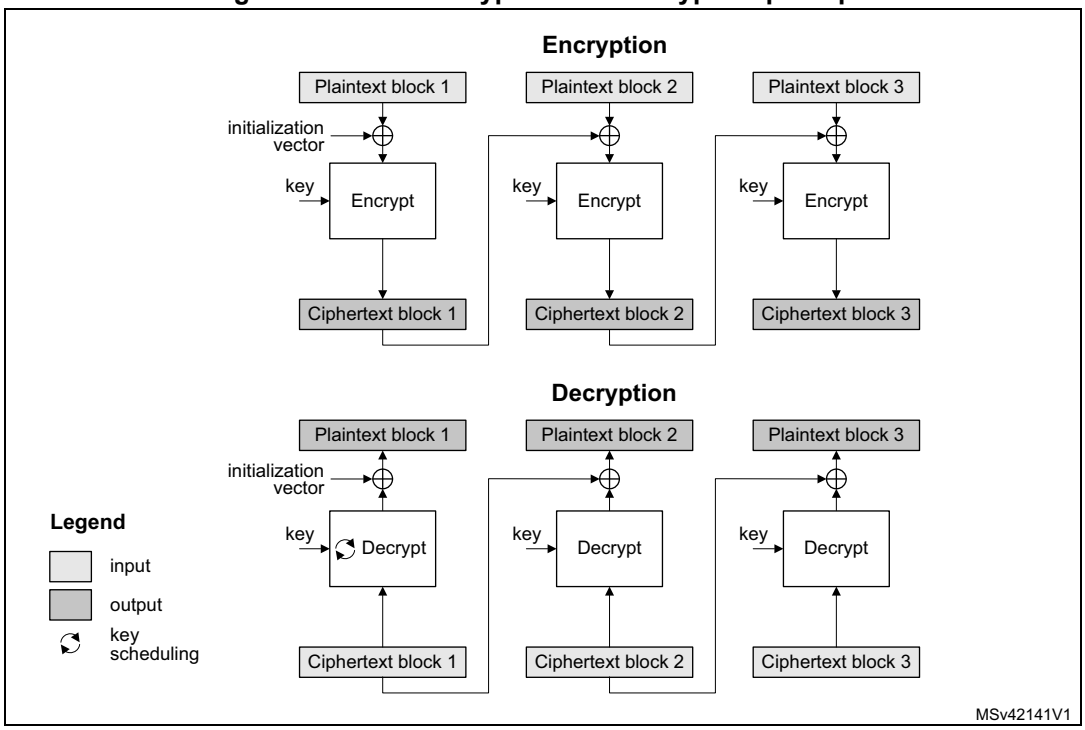


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.

Cipher block chaining (CBC) mode

Figure 196. CBC encryption and decryption principle

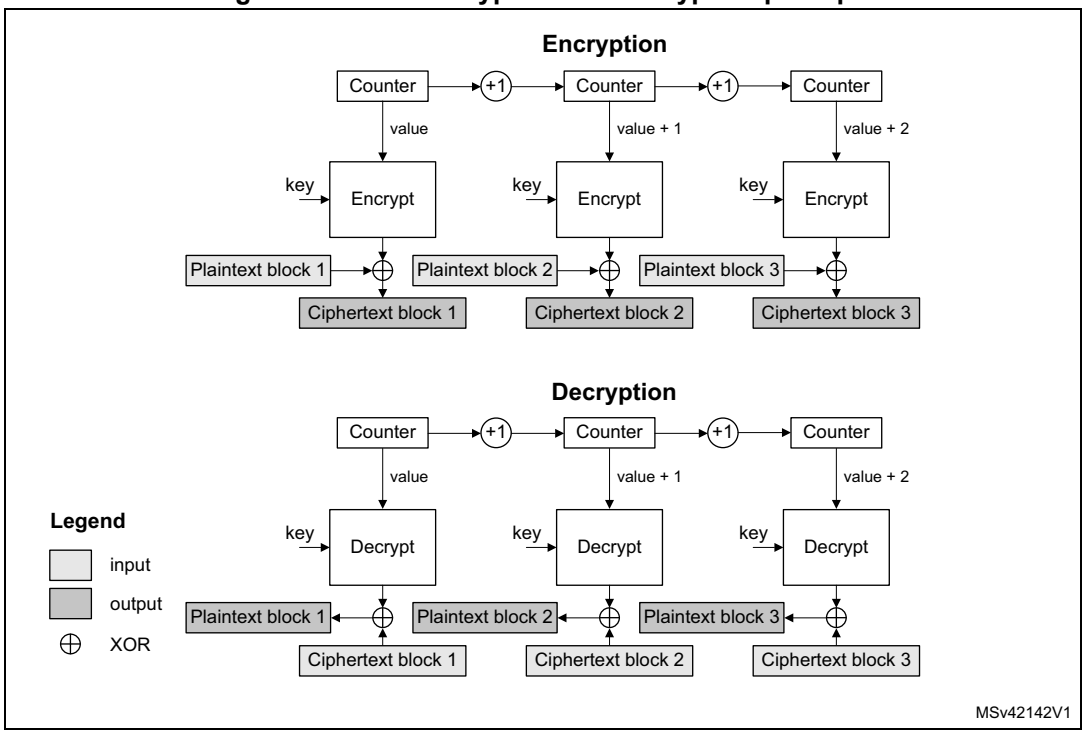


In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

Note: For decryption, a special key scheduling is required before processing the first block.

Counter (CTR) mode

Figure 197. CTR encryption and decryption principle

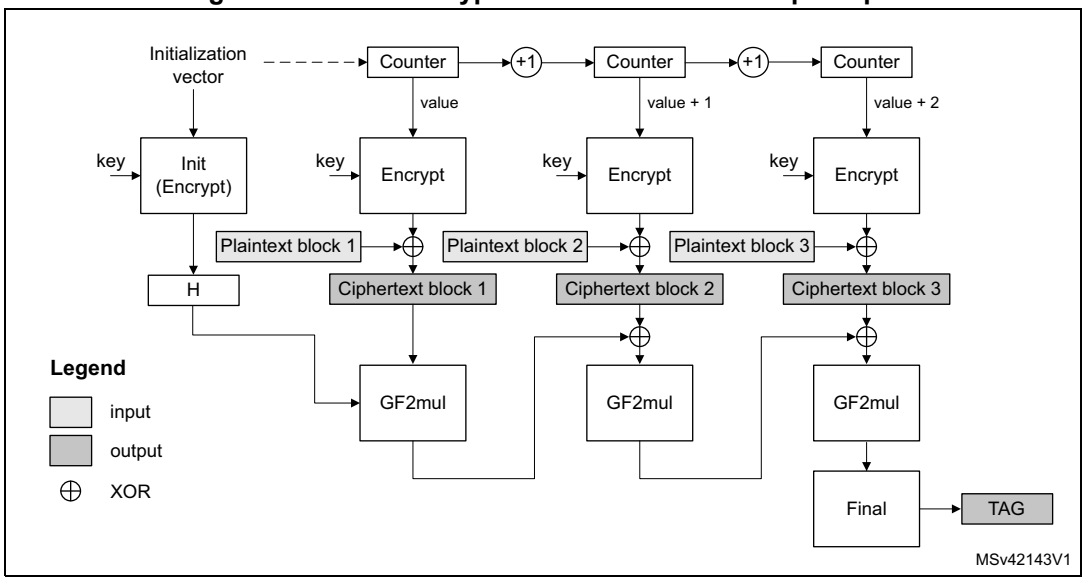


The CTR mode uses the AES core to generate a key stream. The keys are then XORed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

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 198. GCM encryption and authentication principle

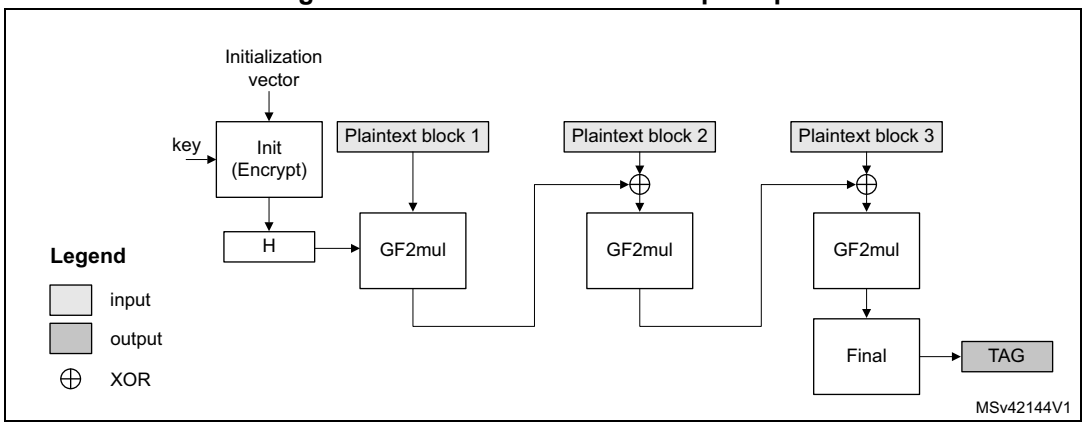


In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block 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 199. GMAC authentication principle

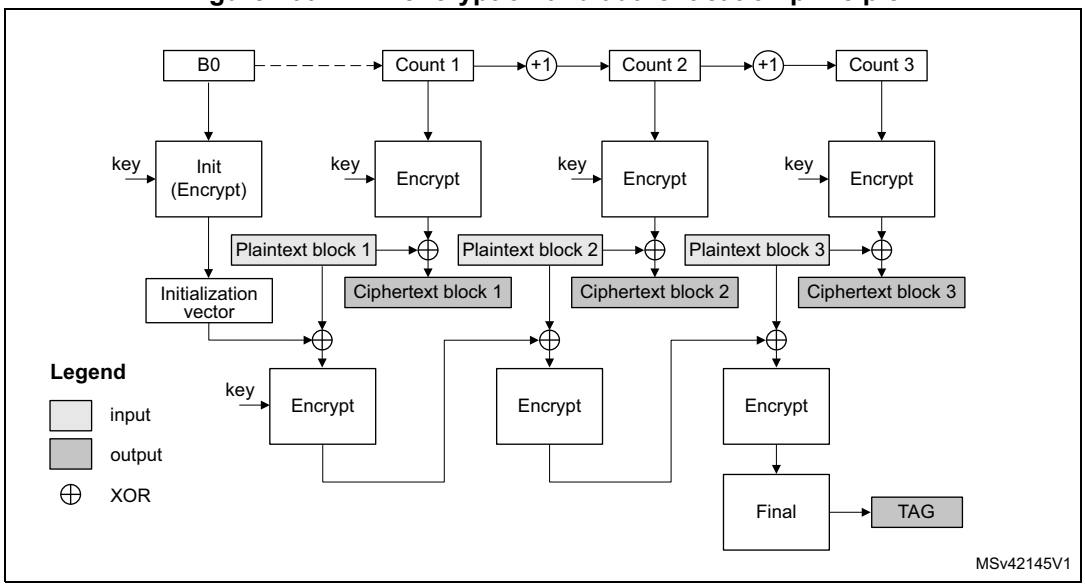


Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

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 200. 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 usage is not recommended by NIST.

29.4.4 AES procedure to perform a cipher operation

Introduction

A typical cipher operation is explained below. Detailed information is provided in sections starting from *Section 29.4.8: 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 29.4.5: 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 29.4.6: 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 29.4.16: 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.

29.4.5 AES decryption round key preparation

Internal key schedule is used to generate AES round keys. In AES encryption, the round 0 key is the one stored in the key registers. AES decryption must start using the last round key. As the encryption key is stored in memory, a special key scheduling must be performed to obtain the decryption key. This key scheduling is only required for AES decryption in ECB and CBC modes.

Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES_CR (key process only), then proceed with the decryption by setting MODE[1:0] to 10 (Mode 3, decryption only). Mode 2 usage is described below:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES_CR. The CHMOD[2:0] bitfield is not significant in this case because this key derivation mode is independent of the chaining algorithm selected.
3. Set key length to 128 or 256 bits, via KEYSIZE bit of AES_CR register.
4. Write the AES_KEYRx registers (128 or 256 bits) with encryption key. Writes to the AES_IVRx registers have no effect.
5. Enable the AES peripheral, by setting the EN bit of the AES_CR register.
6. Wait until the CCF flag is set in the AES_SR register.
7. Clear the CCF flag. Derived key is available in AES core, ready to use for decryption.

Note: The AES is disabled by hardware when the derivation key is available.

To restart a derivation key computation, repeat steps 4, 5, 6, and 7.

Note: The operation of the key preparation lasts 59 or 82 clock cycles, depending on the key size (128- or 256-bit).

29.4.6 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 29.4.4: AES procedure to perform a cipher operation on page 963* 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 29.4.13: AES data registers and data swapping for details).

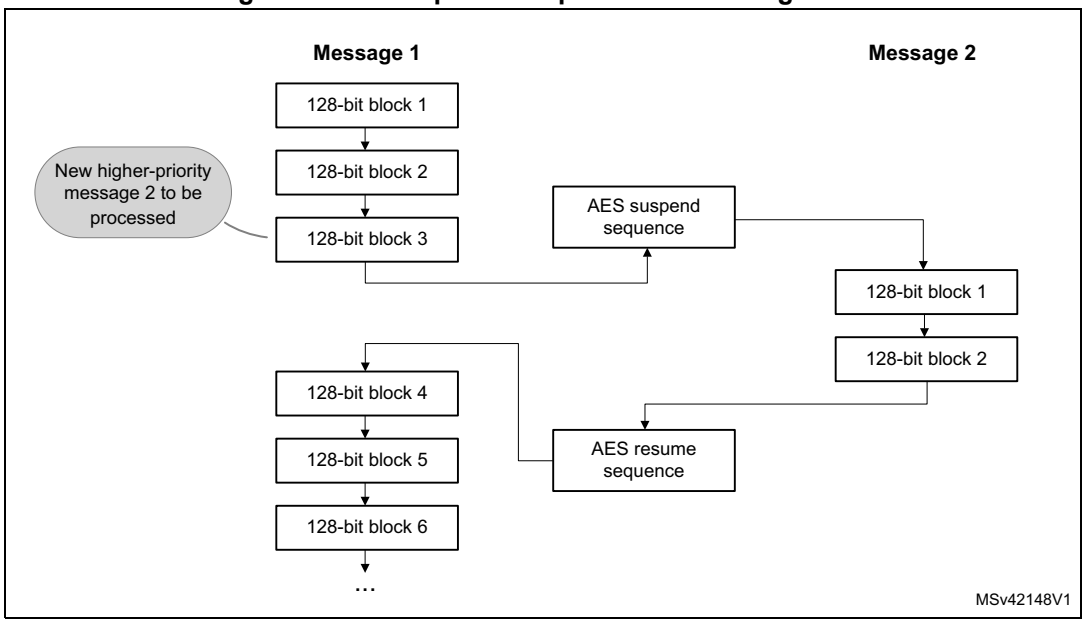
29.4.7 AES task suspend and resume

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 201 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 201. Example of suspend mode management



A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

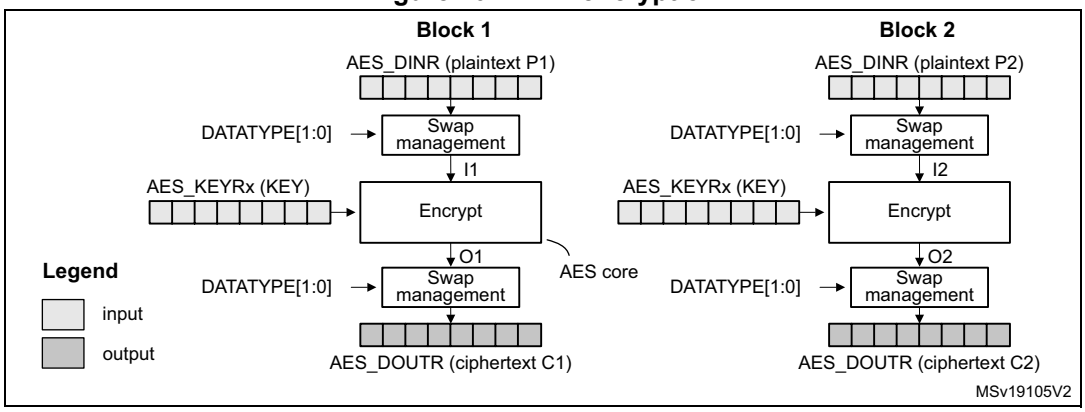
29.4.8 AES basic chaining modes (ECB, CBC)

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 202 illustrates the electronic codebook (ECB) encryption.

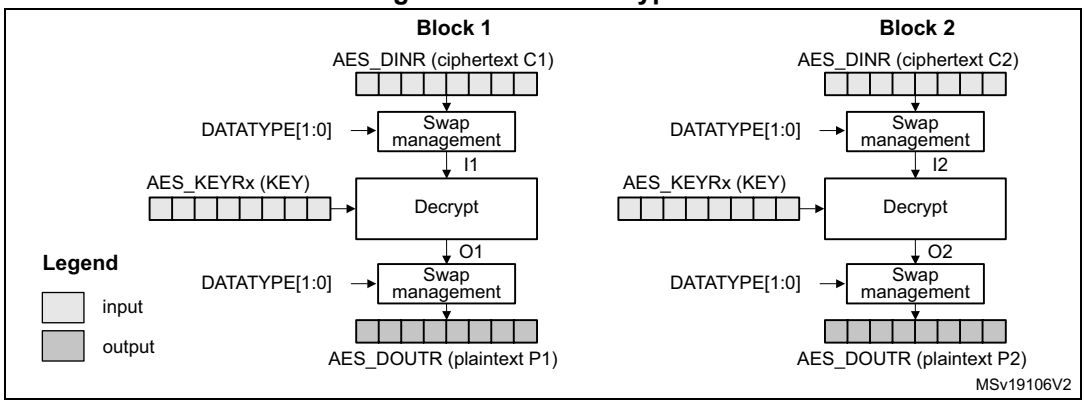
Figure 202. ECB encryption



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 203 illustrates the electronic codebook (ECB) decryption.

Figure 203. ECB decryption

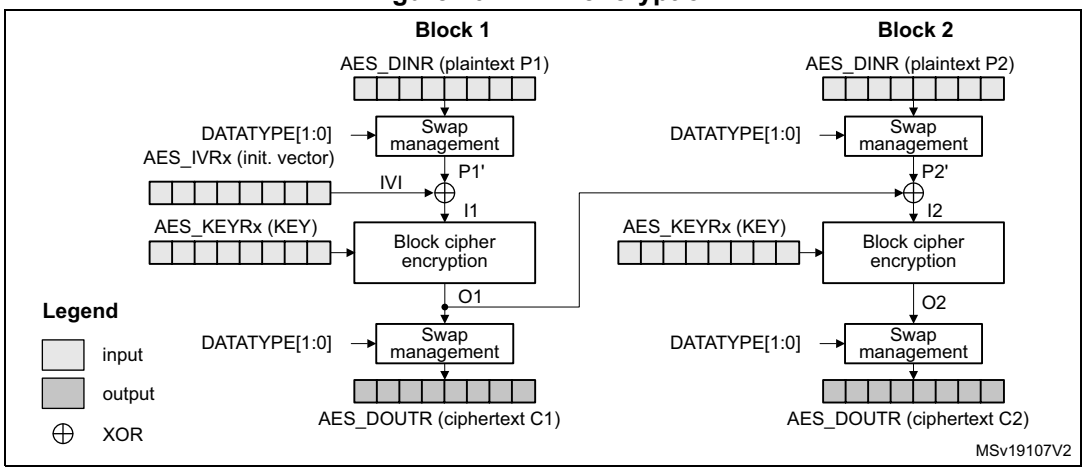


To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported 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 204 illustrates the cipher block chaining (CBC) encryption.

Figure 204. CBC encryption

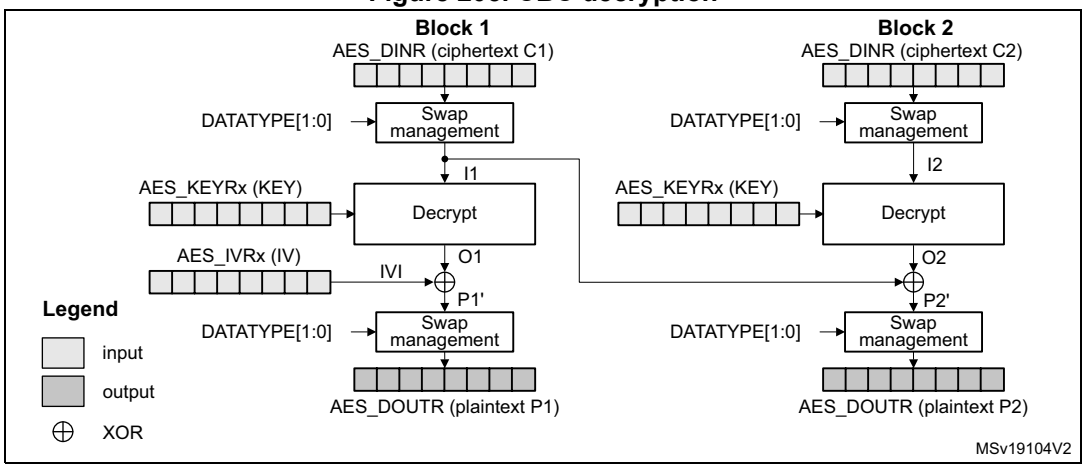


In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after 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 29.4.6: AES ciphertext stealing and data padding*.

Figure 205 illustrates the cipher block chaining (CBC) decryption.

Figure 205. 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 29.4.6: AES ciphertext stealing and data padding*.

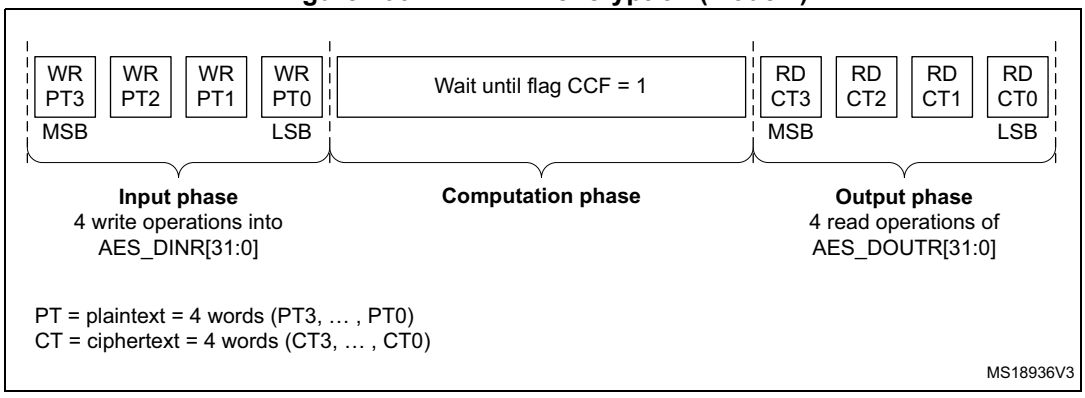
For more information on data swapping, refer to *Section 29.4.13: AES data registers and data swapping*.

ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in *Section 29.4.4*):

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 206*.
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 206*. 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 206. ECB/CBC encryption (Mode 1)



ECB/CBC decryption sequence

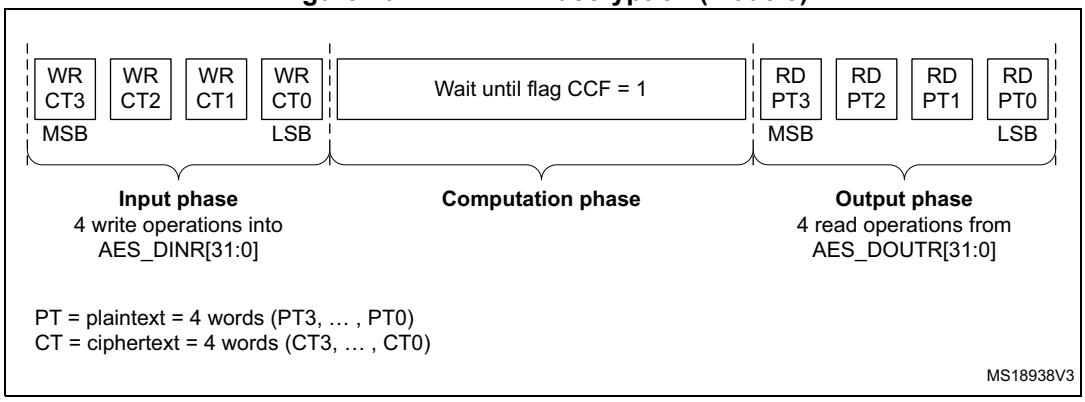
The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in *Section 29.4.4*).

1. Follow the steps described in *Section 29.4.5: 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 207*.
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 207*. 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 207. ECB/CBC decryption (Mode 3)



Suspend/resume operations in ECB/CBC modes

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register.
2. If DMA is not used, read four times the AES_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.
3. If DMA is not used, poll the CCF flag of the AES_SR register until it becomes 1 (computation completed).
4. Clear the CCF flag by setting the CCFC bit of the AES_CR register.
5. Save initialization vector registers (only required in CBC mode as AES_IVRx registers are altered during the data processing).
6. Disable the AES peripheral by clearing the bit EN of the AES_CR register.
7. Save the AES_CR register and clear the key registers if they are not needed, to process the higher priority message.
8. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, and so on).

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Ensure that AES is disabled (the EN bit of the AES_CR must be 0).
3. Restore AES_CR register (with correct KEYSIZE) then restore AES_KEYRx registers.
4. Prepare the decryption key as described in *Section 29.4.5: AES decryption round key preparation* (only required for ECB or CBC decryption).
5. Restore AES_IVRx registers using the saved configuration (only required in CBC mode).
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.
7. If DMA is used, enable AES DMA transfers by setting the DMAINEN and DMAOUTEN bits of the AES_CR register.

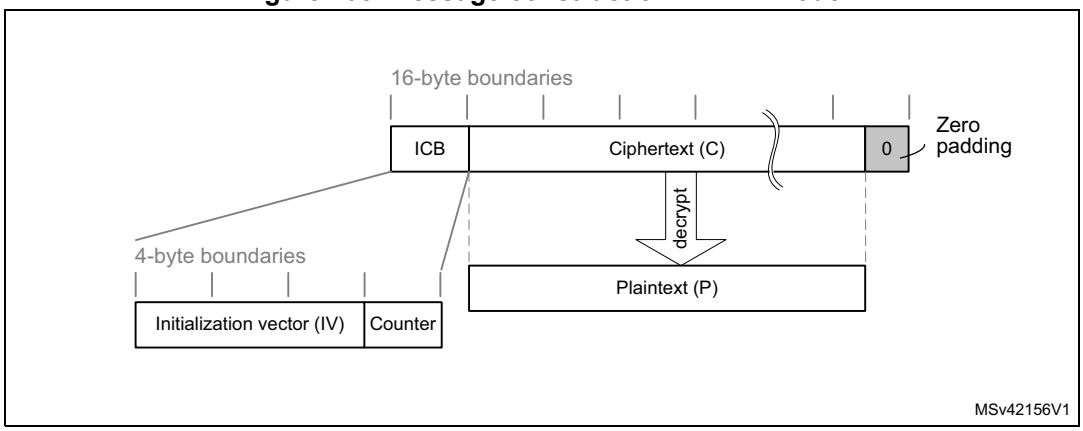
29.4.9 AES counter (CTR) mode

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 208*.

Figure 208. 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 209 and *Figure 210* 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 209. CTR encryption

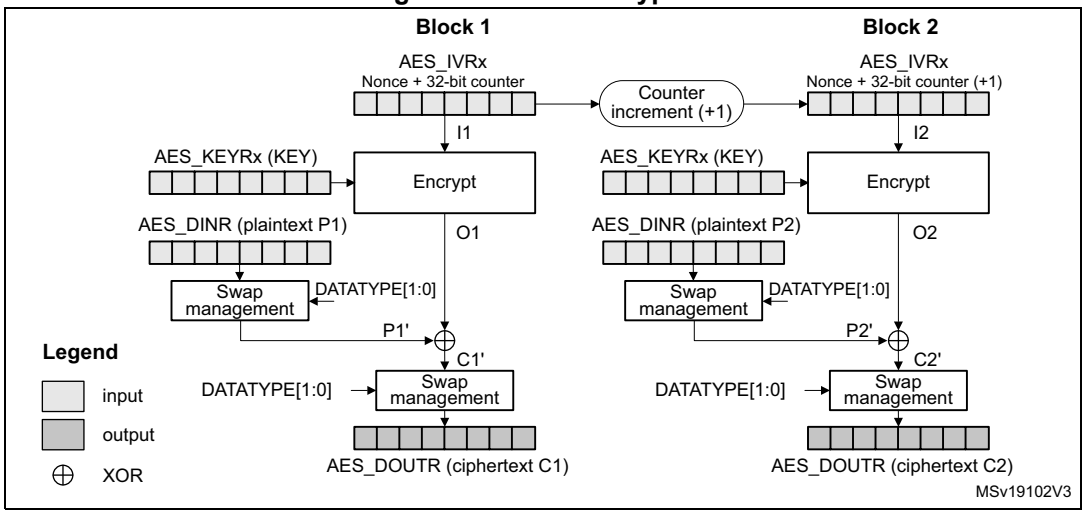
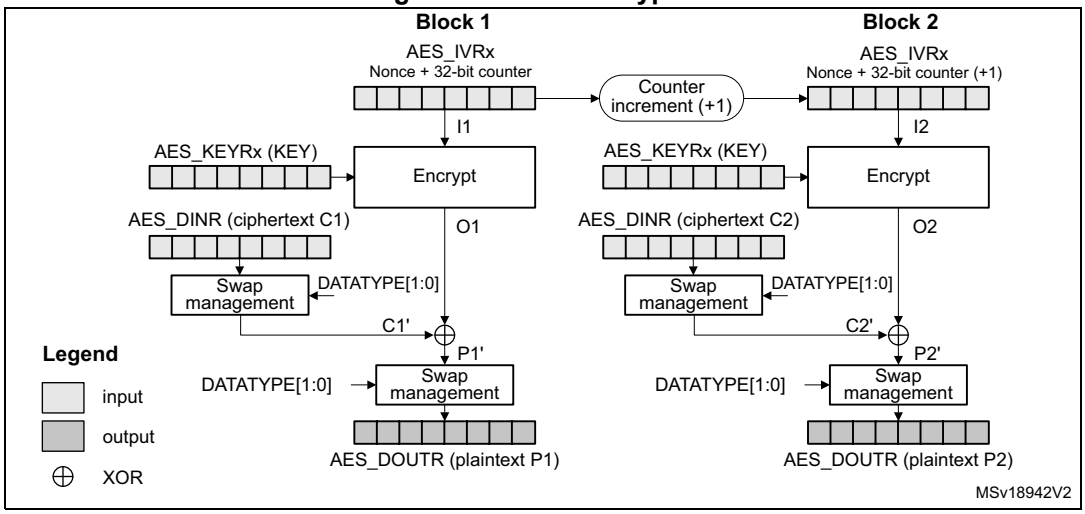


Figure 210. CTR decryption



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 217*.

Table 217. CTR mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
Nonce[31:0]	Nonce[63:32]	Nonce[95:64]	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. Ensure that AES is disabled (the EN bit of the AES_CR must be 0).
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 217*.
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 29.4.4: 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 29.4.8: AES basic chaining modes (ECB, CBC)*.

Note: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.

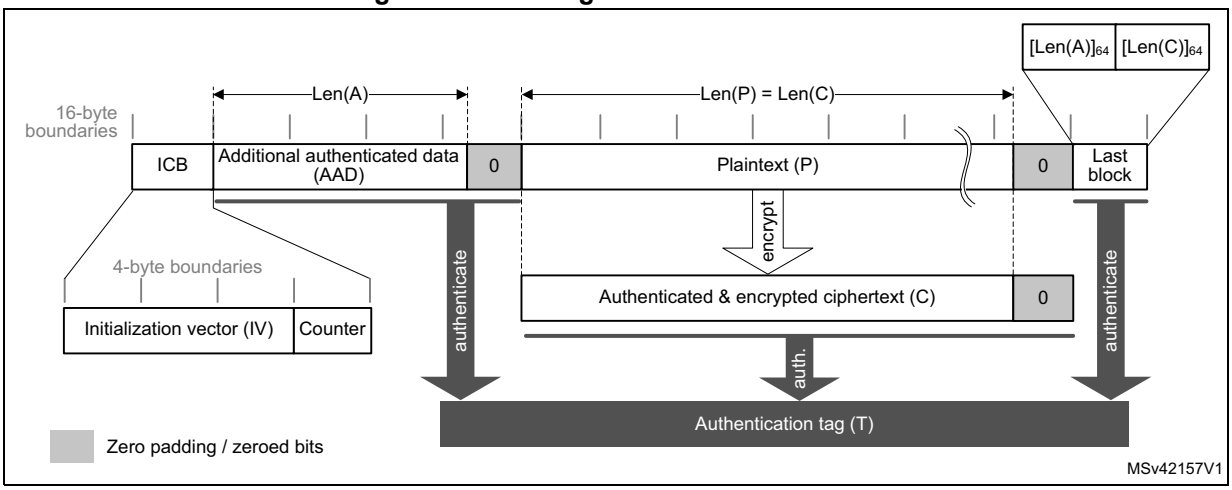
29.4.10 AES Galois/counter mode (GCM)

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 211*.

Figure 211. 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 218*.

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.

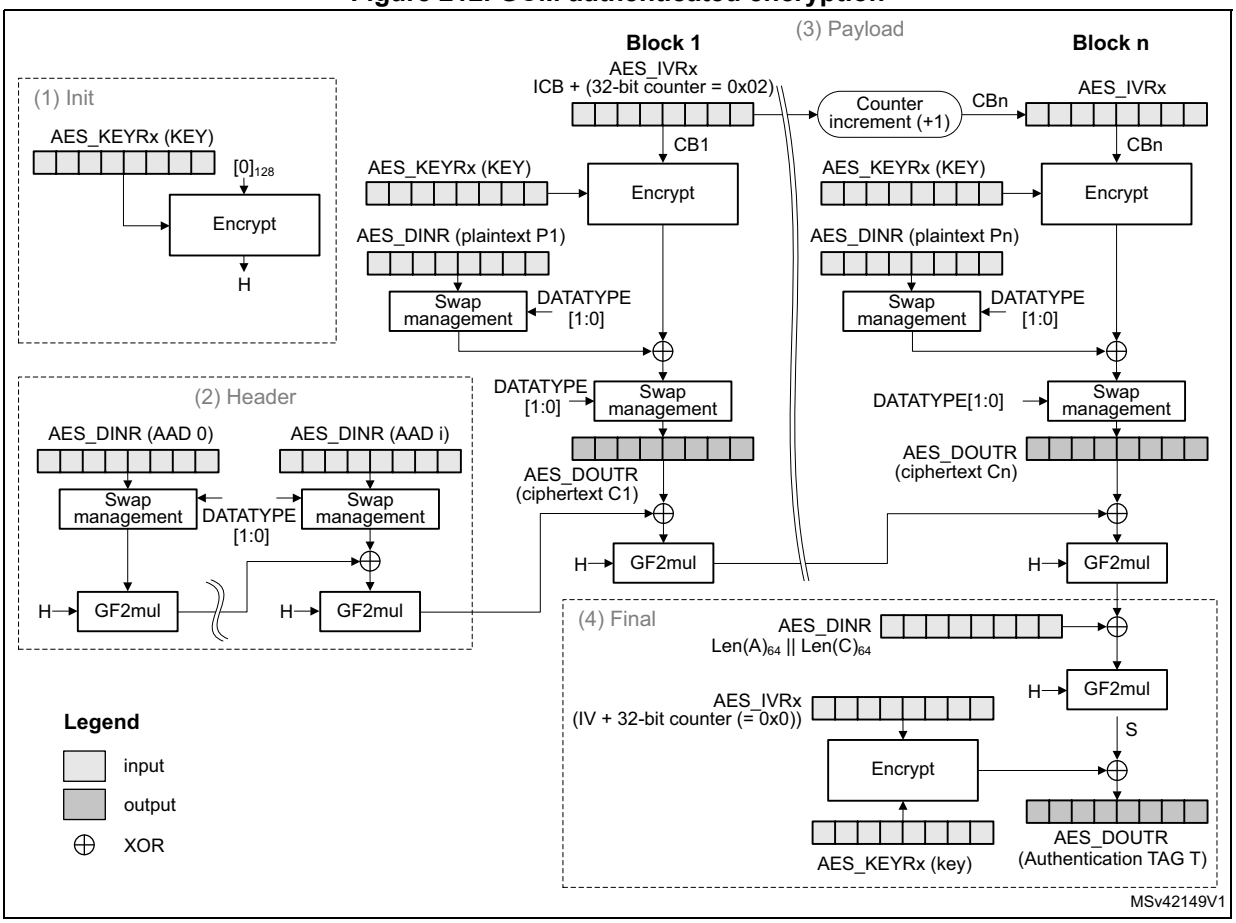
Table 218. GCM last block definition

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 212 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 212. 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 219*).

Table 219. GCM mode IVI bitfield initialization

Register	AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
Input data	ICB[31:0]	ICB[63:32]	ICB[95:64]	Counter[31:0] = 0x2

Note: In this mode, the setting 01 of the MODE[1:0] bitfield (key derivation) is forbidden.

The authentication mechanism in GCM mode is based on a hash function called **GF2mul** that performs multiplication by a fixed parameter, called hash subkey (H), within a binary Galois field.

A GCM message is processed through the following phases, further described in next subsections:

- **Init phase**: AES prepares the GCM hash subkey (H).
- **Header phase**: AES processes the additional authenticated data (AAD), with hash computation only.
- **Payload phase**: AES processes the plaintext (P) with hash computation, counter block encryption and data XOR-ing. It operates in a similar way for ciphertext (C).
- **Final phase**: AES generates the authenticated tag (T) using the last block of the message.

GCM init phase

During this first step, the GCM hash subkey (H) is calculated and saved internally, to be used for processing all the blocks. The recommended sequence is:

1. Ensure that AES is disabled (the EN bit of the AES_CR must be 0).
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 219*.
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 29.4.4: 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 29.4.4: AES procedure to perform a cipher operation on page 963*, 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 218*. 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. Ensure that the AES peripheral is disabled (the EN bit of the AES_CR register must be 0).
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.

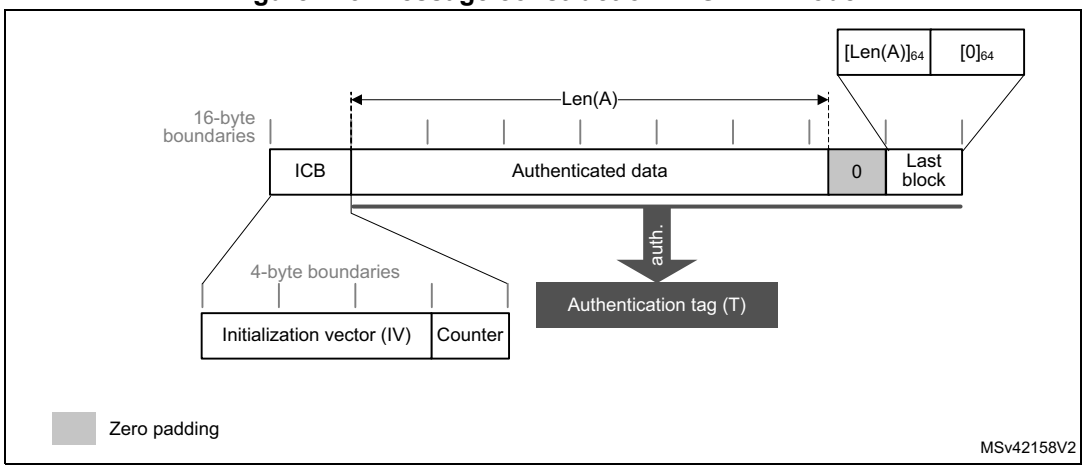
29.4.11 AES Galois message authentication code (GMAC)

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 213*.

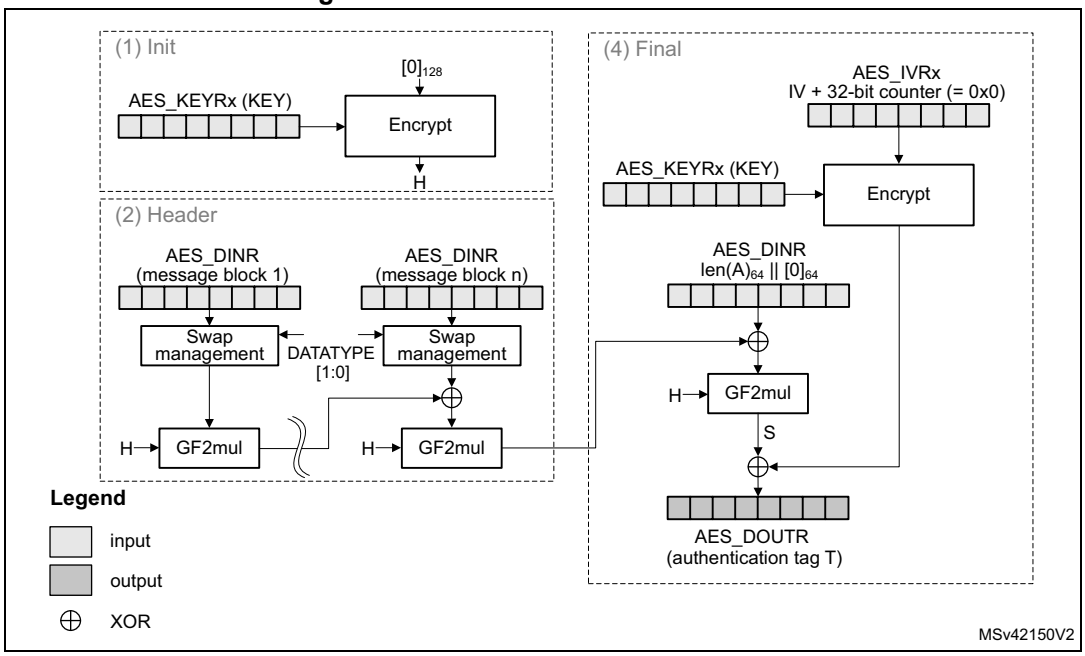
Figure 213. Message construction in GMAC mode



AES GMAC processing

Figure 214 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 214. 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.

29.4.12 AES counter with CBC-MAC (CCM)

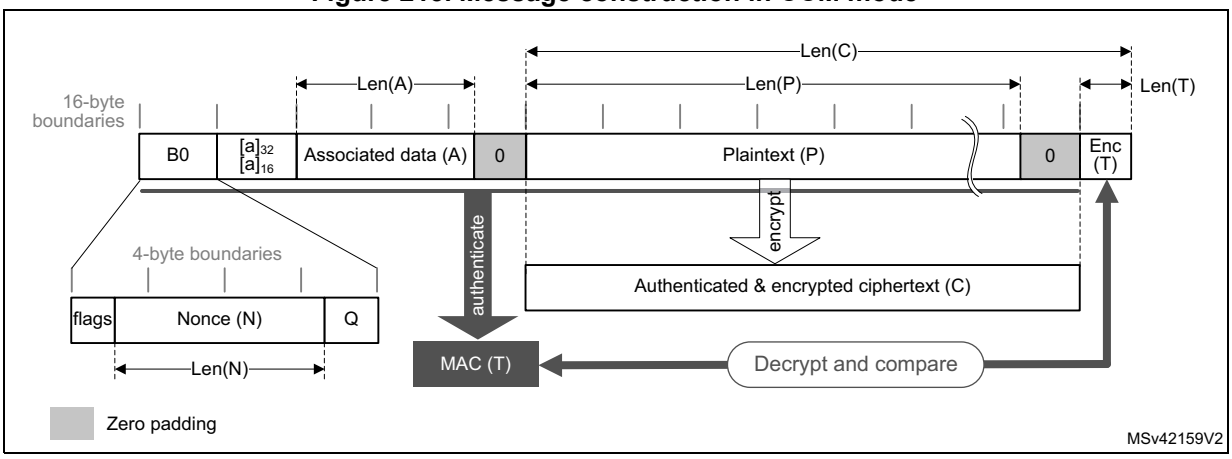
Overview

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 215*.

Figure 215. 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 215*). 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 215*).
- **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

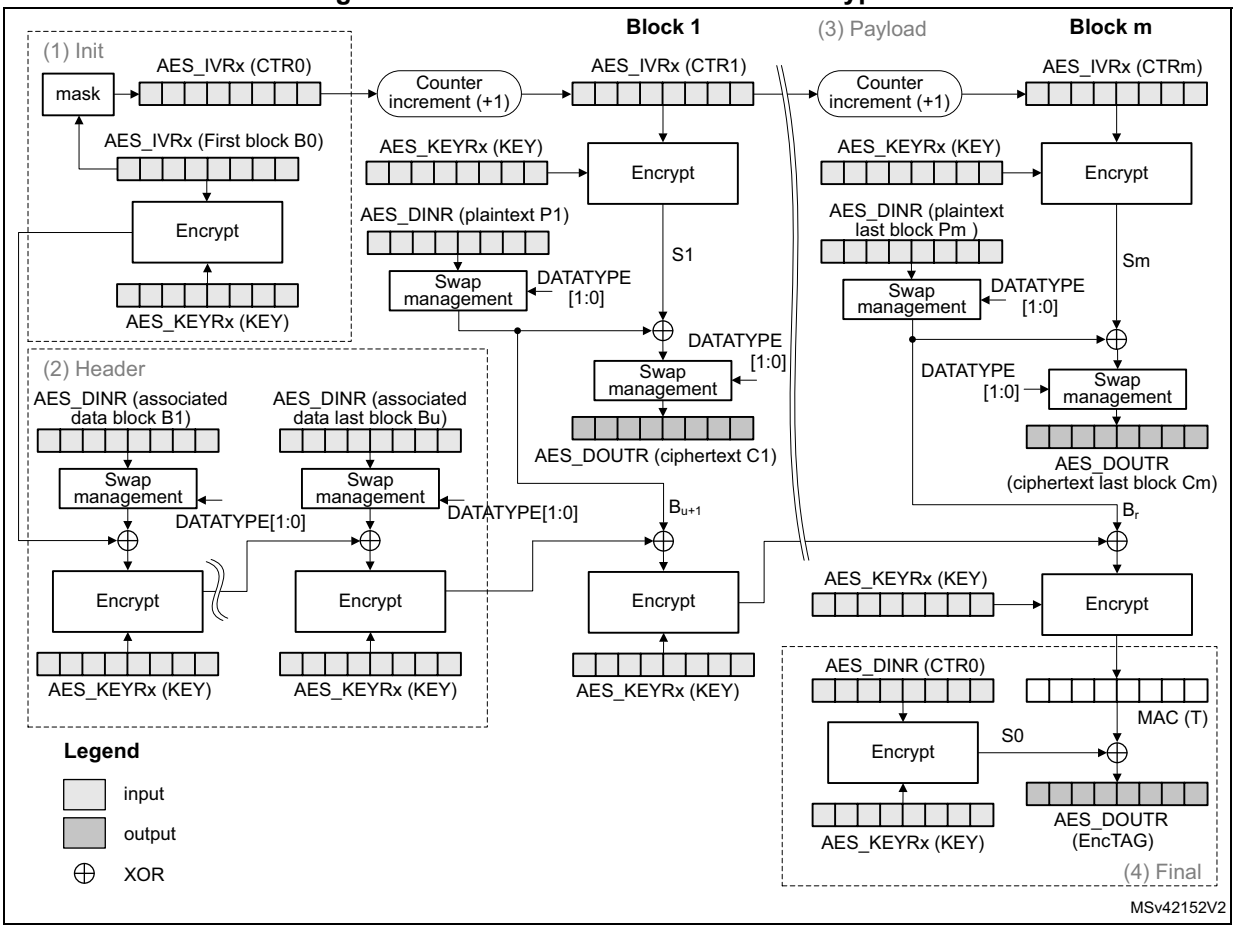
CTR1: 0710111213 141516 00000000 00000001

Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.

CCM processing

Figure 216 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 216. 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 220* 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 220. Initialization of AES_IVRx registers in CCM mode

Register	AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
Input data	B0[31:0]	B0[63:32]	B0[95:64]	B0[127:96]

Note: In this mode, the setting 01 of the MODE[1:0] bitfield (key derivation) is forbidden.

A CCM message is processed through the following phases, further described in next subsections:

- **Init phase**: AES processes the first block and prepares the first counter block.
- **Header phase**: AES processes associated data (A), with tag computation only.
- **Payload phase**: IP processes plaintext (P), with tag computation, counter block encryption, and data XOR-ing. It works in a similar way for ciphertext (C).
- **Final phase**: AES generates the message authentication code (MAC).

CCM Init phase

In this phase, the first block B0 of the CCM message is written into the AES_IVRx register. The AES_DOUTR register does not contain any output data. The recommended sequence is:

1. Ensure that the AES peripheral is disabled (the EN bit of the AES_CR must be 0).
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 220*.
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 29.4.4: 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 29.4.4: AES procedure to perform a cipher operation on page 963*, 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. Ensure that the AES peripheral is disabled (the EN bit of the AES_CR register must be 0).
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.

29.4.13 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 29.4.16: AES DMA interface*.

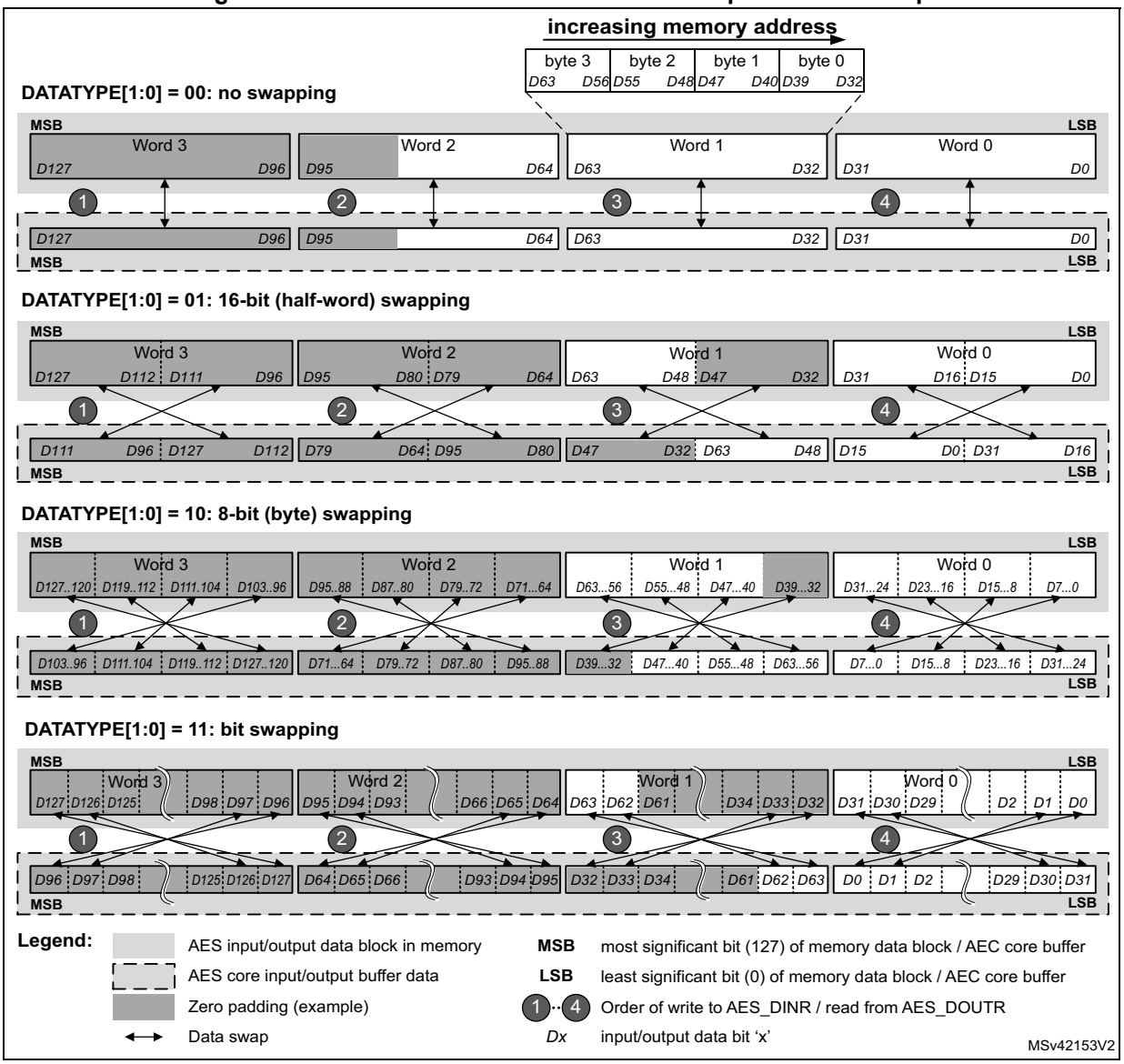
Data swapping

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 217* shows the construction of AES processing core input buffer data P127..0, 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..0.

Figure 217. 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 217 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

29.4.14 AES key registers

The AES_KEYRx write-only registers store the encryption or decryption key bitfield KEY[127:0] or KEY[255:0]. The data to write to each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address (reads are not allowed for security reason).

The key is spread over eight registers as shown in *Table 221*.

Table 221. 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.

29.4.15 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.

29.4.16 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.

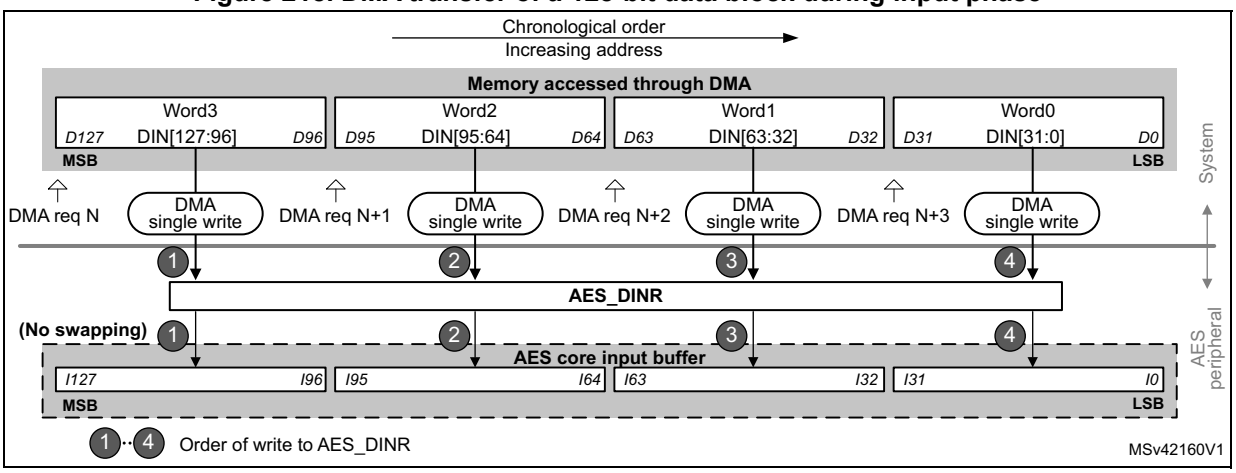
Data input using DMA

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 218*.

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 29.4.4: AES procedure to perform a cipher operation.

Figure 218. DMA transfer of a 128-bit data block during input phase

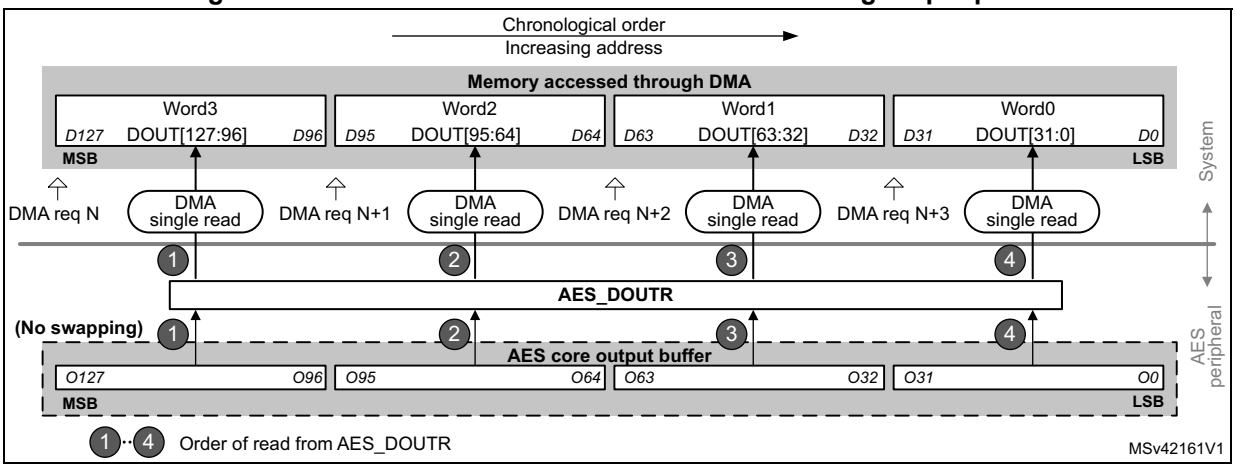


Data output using DMA

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 219*.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Figure 219. 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 29.4.8: AES basic chaining modes (ECB, CBC)* as example.

29.4.17 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 29.5: 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 29.5: AES interrupts*.

The WRERR flag is cleared by setting the ERRC bit of the AES_CR register.

29.5 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 222 gives a summary of the interrupt sources, their event flags and enable bits.

Table 222. 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.

29.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

Table 223. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Algorithm	Clock cycles
128-bit	Mode 1: Encryption	ECB, CBC, CTR	**51**
	Mode 2: Key derivation	-	**59**
	Mode 3: Decryption	ECB, CBC, CTR	**51**
256-bit	Mode 1: Encryption	ECB, CBC, CTR	**75**
	Mode 2: Key derivation	-	**82**
	Mode 3: Decryption	ECB, CBC, CTR	**75**

Table 224. 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).

29.7 AES registers

29.7.1 AES control register (AES_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
								rw	rw	rw	rw		rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

The bitfield sets the number of padding bytes in last block of payload:
0000: All bytes are valid (no padding)
0001: Padding for one least-significant byte of last block
...
1111: Padding for 15 least-significant bytes of last block

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the length of the key used in the AES cryptographic core, in bits:
0: 128
1: 256
Attempts to write the bit are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]:** GCM or CCM phase selection

This bitfield selects the phase of GCM, GMAC or CCM algorithm:
00: Init phase
01: Header phase
10: Payload phase
11: Final phase
The bitfield has no effect if other than GCM, GMAC or CCM algorithms are selected (through the ALGOMODE bitfield).

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables/disables data transferring with DMA, in the output phase:
0: Disable
1: Enable
When the bit is set, DMA requests are automatically generated by AES during the output data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).

Bit 11 **DMAINEN**: DMA input enable

This bit enables/disables data transferring with DMA, in the input phase:
0: Disable
1: Enable
When the bit is set, DMA requests are automatically generated by AES during the input data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).

Bit 10 **ERRIE**: Error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RDERR and/or WRERR is set:
0: Disable (mask)
1: Enable

Bit 9 **CCFIE**: CCF interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set:
0: Disable (mask)
1: Enable

Bit 8 **ERRC**: Error flag clear

Upon written to 1, this bit clears the RDERR and WRERR error flags in the AES_SR register:
0: No effect
1: Clear RDERR and WRERR flags
Reading the flag always returns zero.

Bit 7 **CCFC:** Computation complete flag clear

Upon written to 1, this bit clears the computation complete flag (CCF) in the AES_SR register:
0: No effect
1: Clear CCF
Reading the flag always returns zero.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode selection

This bitfield selects the AES chaining mode:
000: Electronic codebook (ECB)
001: Cipher-block chaining (CBC)
010: Counter mode (CTR)
011: Galois counter mode (GCM) and Galois message authentication code (GMAC)
100: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: AES operating mode

This bitfield selects the AES operating mode:
00: Mode 1: encryption
01: Mode 2: key derivation (or key preparation for ECB/CBC decryption)
10: Mode 3: decryption
11: reserved
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type selection

This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register, through selecting the mode of data swapping:
00: None
01: Half-word (16-bit)
10: Byte (8-bit)
11: Bit
For more details, refer to *Section 29.4.13: 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.

29.7.2 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.

29.7.3 AES data input register (AES_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

Only 32-bit access type is supported.

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

Bits 31:0 **DIN[31:0]**: Input data word

A four-fold sequential write to this bitfield during the input phase results in writing a complete 128-bit block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 128-bit input buffer.

The data signification of the input data block depends on the AES operating mode:

- **Mode 1** (encryption): plaintext
- **Mode 2** (key derivation): the bitfield is not used (AES_KEYRx registers used for input)
- **Mode 3** (decryption): ciphertext

The data swap operation is described in *Section 29.4.13: AES data registers and data swapping on page 987*.

29.7.4 AES data output register (AES_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

Only 32-bit read access type is supported.

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

Bits 31:0 **DOUT[31:0]**: Output data word

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF set), virtually reads a complete 128-bit block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

The data signification of the output data block depends on the AES operating mode:

- **Mode 1** (encryption): ciphertext
- **Mode 2** (key derivation): the bitfield is not used
- **Mode 3** (decryption): plaintext

The data swap operation is described in *Section 29.4.13: AES data registers and data swapping on page 987*.

29.7.5 AES key register 0 (AES_KEYR0)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

This write-only bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode:

- In **Mode 1** (encryption), **Mode 2** (key derivation): the value to write into the bitfield is the encryption key.
- In **Mode 3** (decryption): the value to write into the bitfield is the encryption key to be derived before being used for decryption.

The AES_KEYRx registers may be written only when KEYSIZE value is correct and when the AES peripheral is disabled (EN bit of the AES_CR register cleared). Note that, if, the key is directly loaded to AES_KEYRx registers (hence writes to key register is ignored and KEIF is set).

Refer to *Section 29.4.14: AES key registers on page 989* for more details.

29.7.6 AES key register 1 (AES_KEYR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

29.7.7 AES key register 2 (AES_KEYR2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

29.7.8 AES key register 3 (AES_KEYR3)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

29.7.9 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 29.4.15: AES initialization vector registers on page 989* 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

29.7.10 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.

29.7.11 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.

29.7.12 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.

29.7.13 AES key register 4 (AES_KEYR4)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

29.7.14 AES key register 5 (AES_KEYR5)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

29.7.15 AES key register 6 (AES_KEYR6)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

29.7.16 AES key register 7 (AES_KEYR7)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

Note: The key registers from 4 to 7 are used only when the key length of 256 bits is selected. They have no effect when the key length of 128 bits is selected (only key registers 0 to 3 are used in that case).

29.7.17 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.

29.7.18 AES register map

Table 225. AES register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	AES_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	WRERR	RDERR	CCF
	Reset value																													0	0	0	0
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 225. 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
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 2.3 on page 87* for the register boundary addresses.

30 Hash processor (HASH)

30.1 Introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-224, SHA-256), the MD5 (message-digest algorithm 5) hash algorithm and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of up to (2^{64} – 1) bits. It also computes 128-bit digests for the MD5 algorithm.

30.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 1321, *MD5 Message-Digest Algorithm*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Fast computation of SHA-1, SHA-224, SHA-256, and MD5
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 66 clock cycles for processing one 512-bit block of data using MD5 algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Word swapping supported: bits, bytes, half-words and 32-bit words
- Automatic padding to complete the input bit string to fit digest minimum block size of 512 bits (16 × 32 bits)
- Single 32-bit input register associated to an internal input FIFO, corresponding to one block size
- AHB slave peripheral, accessible through 32-bit word accesses only (else an AHB error is generated)
- 8 × 32-bit words (H0 to H7) for output message digest
- Automatic data flow control with support of direct memory access (DMA) using one channel.
- Only single DMA transfers are supported

- Interruptible message digest computation, on a per-block basis
  - Re-loadable digest registers
  - Hashing computation suspend/resume mechanism, including DMA

30.3 HASH implementation

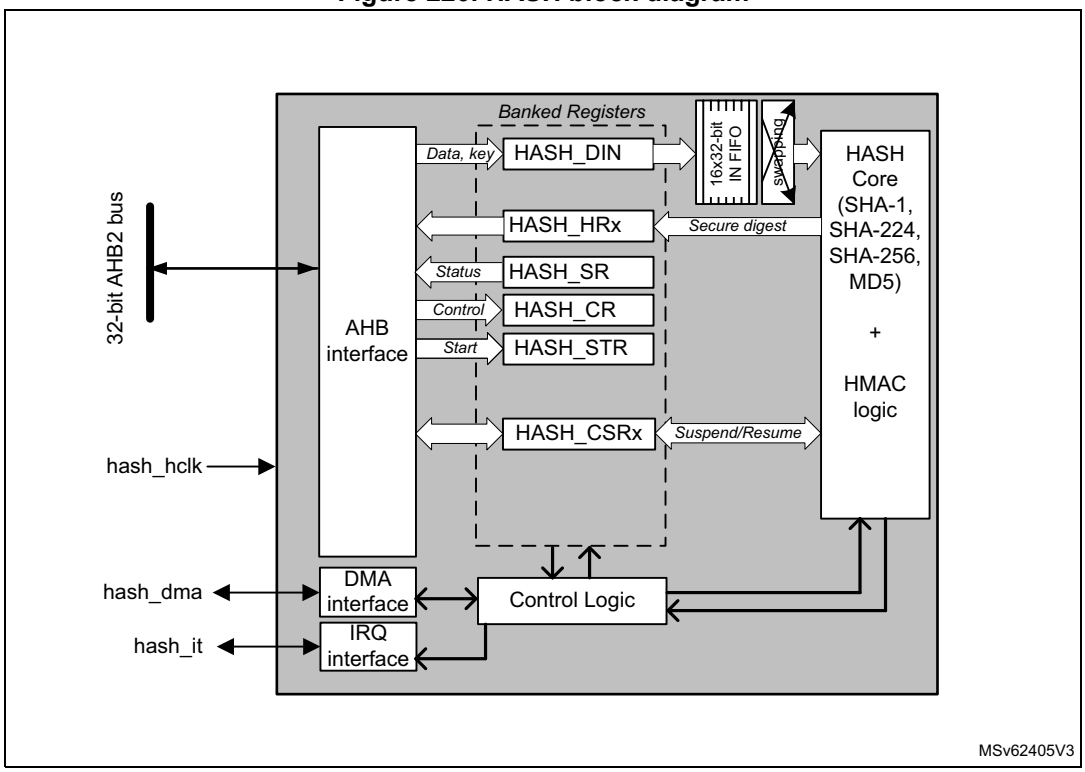
The devices have a single instance of HASH peripheral.

30.4 HASH functional description

30.4.1 HASH block diagram

Figure 220 shows the block diagram of the hash processor.

Figure 220. HASH block diagram



30.4.2 HASH internal signals

Table 226 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 226. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA transfer request/ acknowledge

30.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard and the IETF RFC1321 publication (MD5).

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message of any length below 2^{64} bits is provided on input, the HASH processing core produces respectively a fixed-length output string called a message digest, defined as follows:

- For MD5 digest size is 128-bit
- For SHA-1 digest size is 160-bit
- For SHA-224 and SHA-256, the digest size is 224 bits and 256 bits, respectively

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017). Any change to a message in transit, with very high probability, results in a different message digest, and the signature fails to verify.

30.4.4 Message data feeding

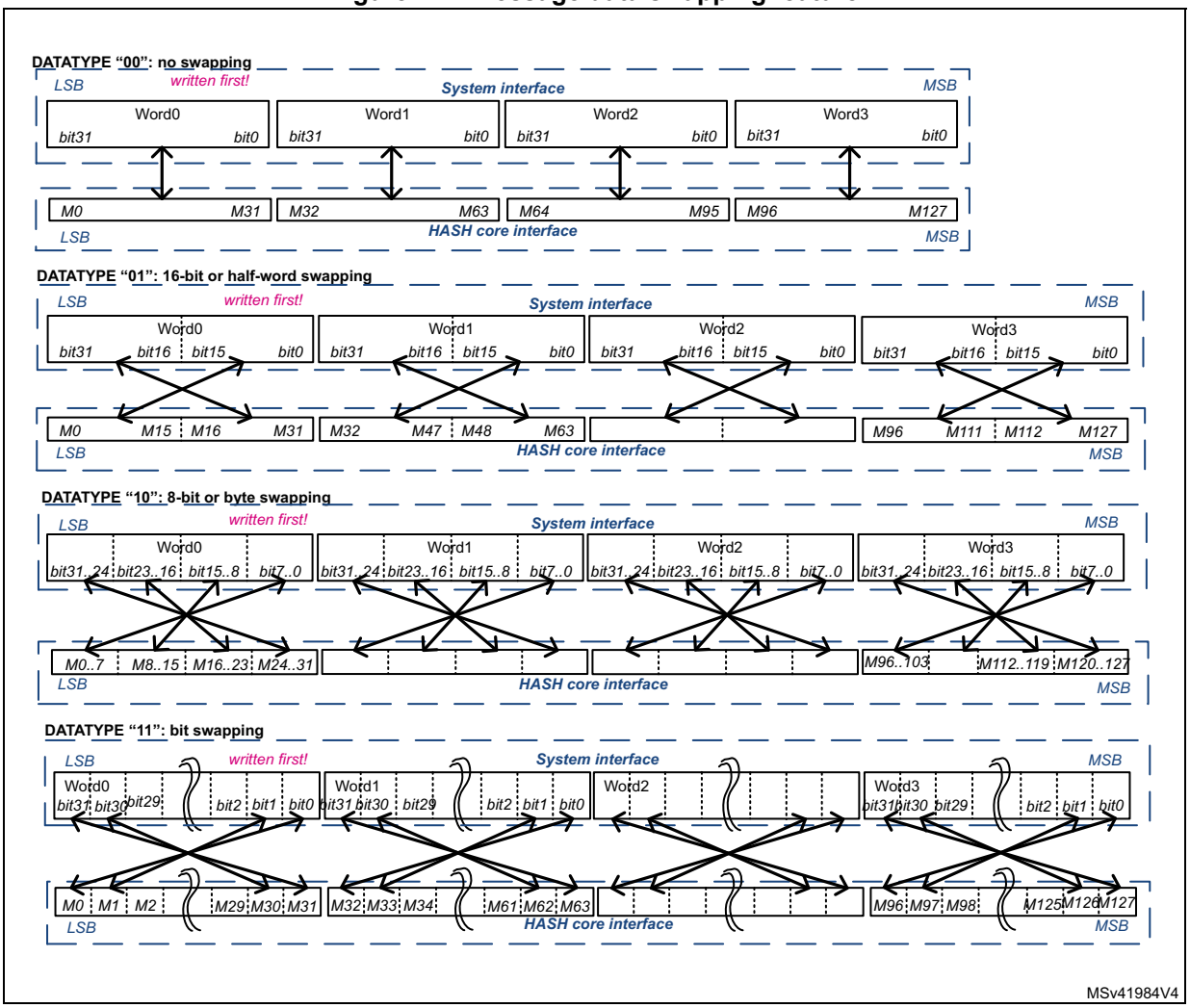
The message (or data file) to be processed by the HASH should be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “`01100001 01100010 01100011`” is represented by a 32-bit word `0x00636261`, and 8-bit words `0x61626300`.

Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (e.g. byte swapping when data are ASCII text stream) there must be a bit, byte, half-word or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 221* shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR).

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 221. Message data swapping feature



30.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. For MD5, SHA1 and SHA2, the block size is 512 bits.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 30.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO.

In order to perform the hash computation on this data below sequence must be used by the application:

1. Initialize the hash processor using the HASH_CR register:
   a) Select the right algorithm using the ALGO bitfield. If needed program the correct swapping operation on the message input words using DATATYPE bitfield in HASH_CR.
   b) When the HMAC mode is required, set the MODE bit, as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (else keep LKEY cleared). Refer to *Section 30.4.7: HMAC operation* for details.
   c) Set MODE = 1 and select the key length using LKEY if HMAC mode has been selected.
   d) Update NBLW[4:0] to define the number of valid bits in last word of the message if it is different from 32 bits. NBLW[4:0] information are used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting to 1 the INIT bit in HASH_CR. Also set the bit DMAE to 1 if data are transferred via DMA.

Caution: When programming step 2, it is important to set up before or at the same time the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW[4:0]).

3. Start filling data by writing to HASH_DIN register, unless data are automatically: transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   a) When data are filled by software:
   – Partial digest computation are triggered each time the application writes the first word of the next block. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.
   – The final digest computation is triggered when the last block is entered and the software writes the DCAL bit to 1. If the message length is not an exact multiple of the block size, the NBLW[4:0] bitfield in HASH_STR register must be written prior to writing DCAL bit (see *Section 30.4.6* for details).
   b) When data are filled by DMA as a single DMA transfer (MDMAT bit = 0):
   – Partial digest computations are triggered automatically each time the FIFO is full.
   The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register (DCAL bit is set to 1 by hardware). If the message length is not an exact multiple of the block size, the NBLW[4:0] field

in HASH_STR register must be written prior to enabling the DMA (see *Section 30.4.6* for details).

c) When data are filled using multiple DMA transfers (MDMAT bit = 1):

– Partial digest computations are triggered as for single DMA transfers. However the final digest computation is not triggered automatically when the last block has been transferred to the HASH_DIN register (DCAL bit is not set to 1 by hardware). It allows the hash processor to receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must set MDMAT bit to 0 before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers (see description before).

4. Once the digest computation is complete (DCIS = 1), the resulting digest can be read from the output registers as described in *Table 227*.

Table 227. Hash processor outputs

Algorithm	Valid output registers	Most significant bit	Digest size (in bits)
MD5	HASH_H0 to HASH_H3	HASH_H0[31]	128
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA-256	HASH_H0 to HASH_H7		256

For more information about HMAC detailed instructions, refer to *Section 30.4.7: HMAC operation*.

30.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW[4:0], has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed padding sequences with DMA enabled or disabled are described in *Section 30.4.5: Message digest computing*.

Padding example

As specified by Federal Information Processing Standards PUB 180-4, the message padding consists in appending a “1” followed by k “0”s, itself followed by a 64-bit integer that is equal to the length L in bits of the message. These three padding operations generate a padded message of length $L + 1 + k + 64$, which by construction is a multiple of 512 bits.

For the hash processor, the “1” is added to the last word written to the HASH_DIN register at the bit position defined by the NBLW[4:0] bitfield, and the remaining upper bits are cleared (“0”s).

Example from FIPS PUB180-4

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24:

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW[4:0] has to be loaded with the value 24: a **“1”** is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

The message length value, L, in two-word format (that is 00000000 00000018) is appended. Hence the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered by following the below sequence:

1. **0xUU636261** is written to the HASH_DIN register (where ‘U’ means don’t care).
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits).
3. **0x10** is written to the HASH_STR register to start the message padding (described above) and then perform the digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0...4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```
HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D
```

30.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

```
HMAC(message)= Hash((Key | pad) XOR opad |
               Hash((Key | pad) XOR ipad | message))
```

where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the size of the underlying hash function data block (n = 64 for 512-bit blocks).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n,* the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software writes the INIT bit to 1 with the MODE bit at 1 and the ALGO bits set to the value corresponding to the desired algorithm. The LKEY bit must also be set to 1 if the key being used is longer than 64 bytes. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading, that is writing the key data into HASH_DIN register then completing the transfer by writing DCAL bit to 1 and the correct NBLW[4:0] to HASH_STR register.

Note: Endianness details can be found in Section 30.4.4: Message data feeding.

3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software writes DCAL bit to 1 in HASH_STR register, the NBLW[4:0] bitfield must be written at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 30.4.4: Message data feeding*.
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register then completing the transfer by writing DCAL bit to 1 and the correct NBLW[4:0] to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HR7) as soon as DCIS bit is set to 1.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 30.6: HASH processing time.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST.

Let us assume that the original message is the ASCII binary-coded form of "**Sample message for keylen = blocklen**", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is no padding), specified by NIST. As key length = 64, LKEY bit is set to 0 in HASH_CR register

   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617

   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F

   30313233 34353637 38393A3B 3C3D3E3F

2. **Message** input (length = 34, that is padding required). HASH_STR must be set to **0x20** to start message padding and inner hash computation (see 'U' as don't care)

   53616D70 6C65206D 65737361 67652066 6F72206B 65796C65

   6E3D626C 6F636B6C 656EUUUU

3. **Outer hash key** input (length = 64, that is no padding). A key identical to the inner hash key is entered here.

4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers (x = 0 to 4), as shown below:

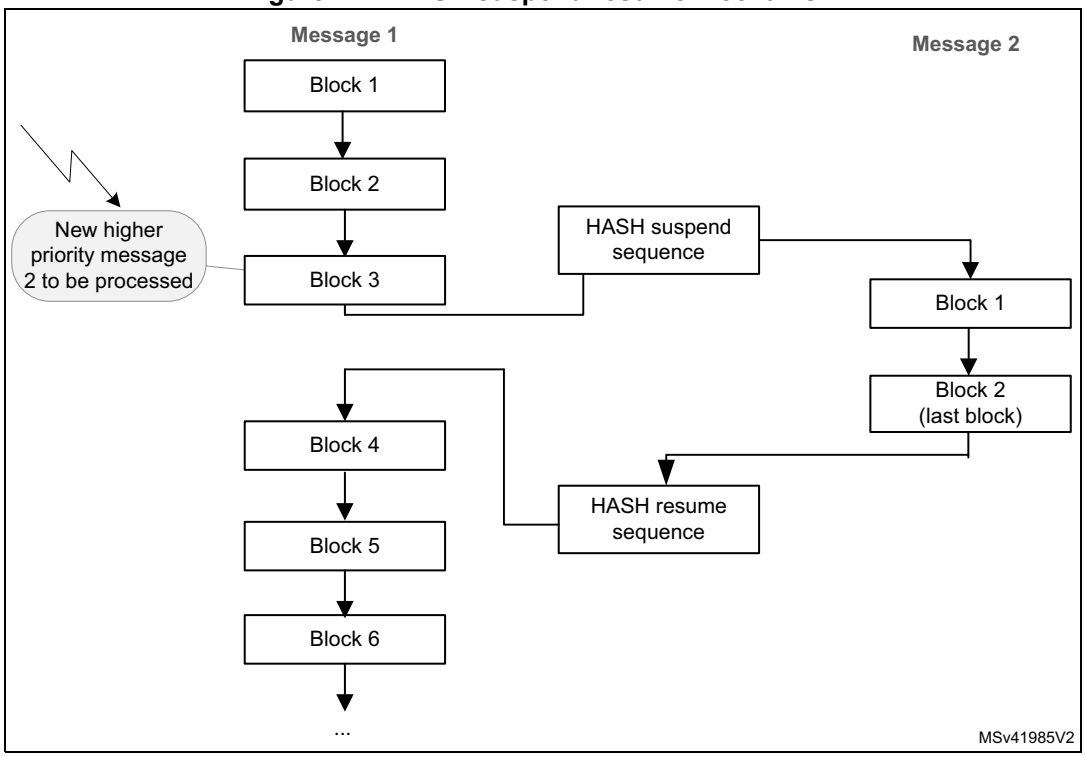
   HASH_HR0 = 0x5FD596EE
   HASH_HR1 = 0x78D5553C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29

30.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 222*.

Figure 222. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described below.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing 16 words 32-bit (plus one if it is the first block):

1. In Polling mode, wait for BUSY = 0, then poll if the DINIS status bit is set to 1.

   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37. HASH_CSR38 to HASH_CSR53 registers must also be saved if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR and HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Note: To optimize the resume process when NBW[3:0] = 0x0, HASH_CSR22 to HASH_CSR37 registers do not need to be saved then restored as the FIFO is empty.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing is described below.

To suspend the processing of a message using DMA, proceed as follows:

1. In Polling mode, wait for BUSY = 0. If DCIS is set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 2.
2. In Polling mode, wait for BUSY = 1.
3. Disable the DMA channel. Then clear DMAE bit in HASH_CR register.
4. In Polling mode, wait for BUSY = 0. If DCIS is set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR, HASH_CR, and HASH_CSR0 to HASH_CSR37 registers. HASH_CSR38 to HASH_CSR53 registers must also be saved if an HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again. Do not forget to take into account the words that have been already pushed into the FIFO if NBW[3:0] is higher than 0x0.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

Note: To optimize the resume process when NBW[3:0] = 0x0, HASH_CSR22 to HASH_CSR37 registers do not need to be saved then restored as the FIFO is empty.

30.4.9 HASH DMA interface

The HASH only supports single DMA transfers.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information refer to *Section 30.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW[4:0] = Len(Message)% 32` where "`x%32`" gives the remainder of x divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

30.4.10 HASH error management

No error flags are generated by the hash processor.

30.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the NVIC (nested vectored interrupt controller). Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 228* gives a summary of the available features.

Table 228. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

30.6 HASH processing time

Table 229 summarizes the time required to process an intermediate block for each mode of operation.

Table 229. Processing time (in clock cycle)

Mode of operation	FIFO load(1)	Computation phase	Total
MD5	16	50	**66**
SHA-1	16	66	**82**
SHA-224	16	50	**66**
SHA-256			

1. Add the time required to load the block into the processor.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

30.7 HASH registers

The HASH core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

30.7.1 HASH control register (HASH_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO1	Res.	LKEY
													rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				ALGO0	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r	rw	rw	rw	rw	rw	rw		

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

Bit 17 Reserved, must be kept at reset value.

Bit 16 **LKEY:** Long key selection

This bit selects between short key (≤ 64 bytes) or long key (> 64 bytes) in HMAC mode.

0: the HMAC key is shorter or equal to 64 bytes. The actual key value written to HASH_DIN is used during the HMAC computation.

1: the HMAC key is longer than 64 bytes. The hash of the key is used instead of the real key during the HMAC computation.

This selection is only taken into account when the INIT bit is set and MODE = 1. Changing this bit during a computation has no effect.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT:** Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE:** DIN not empty

This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is written to 1.

0: No data are present in the data input buffer

1: The input buffer contains at least one word of data

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

This bitfield reflects the number of words in the message that have already been pushed into the IN FIFO. NBW is incremented by one when a write access to the HASH_DIN register is performed (except if DINNE = 0 and the DMA is not used, see below description). NBW goes to zero when the INIT bit is written to 1.

This bitfield is read-only.

If the DMA is not used

0000: if DINNE = 0, no word has been pushed into the DIN buffer (both HASH_DIN register and IN FIFO are empty), otherwise one word has been pushed into the DIN buffer (HASH_DIN register contains one word and IN FIFO is empty)

0001: two words have been pushed into the DIN buffer (that is HASH_DIN register and the IN FIFO contain one word each)

...

1111: 16 words have been pushed into the DIN buffer.

If the DMA is used

NBW contains the exact number of words that have been pushed into the IN FIFO by the DMA.

Bits 18, 7 **ALGO[1:0]**: Algorithm selection

These bits select the hash algorithm.

00: SHA-1

01: MD5

10: SHA-224

11: SHA-256

This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect.

Bit 6 **MODE:** Mode selection

This bit selects the HASH or HMAC mode for the selected algorithm:

0: Hash mode selected

1: HMAC mode selected. LKEY must be set if the key being used is longer than 64 bytes.

This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

Defines the format of the data entered into the HASH_DIN register:

00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.

01: 16-bit data, or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.

10: 8-bit data, or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.

11: bit data, or bit-string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

0: DMA transfers disabled

1: DMA transfers enabled. A DMA request is sent as soon as the HASH core is ready to receive data.

After this bit is set it is cleared by hardware while the last data of the message is written into the hash processor.

Setting this bit to 0 while a DMA transfer is on-going is not aborting this current transfer. Instead, the DMA interface of the IP remains internally enabled until the transfer is completed or INIT is written to 1.

Setting INIT bit to 1 does not clear DMAE bit.

Bit 2 **INIT:** Initialize message digest calculation

Writing this bit to 1 resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.

Writing this bit to 0 has no effect. Reading this bit always return 0.

Bits 1:0 Reserved, must be kept at reset value.

30.7.2 HASH data input register (HASH_DIN)

Address offset: 0x04

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks. When the HASH_DIN register is programmed, the value presented on the AHB databus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB databus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing new data into the HASH_DIN register (the first word of the next block) if the DMA is not used (intermediate digest calculation),
- automatically if the DMA is used.

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by writing the DCAL bit to 1 in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is set to 0.

Reading the HASH_DIN register returns the last word written to this location (zero after reset).

Note: When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

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

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the IN FIFO, and the register takes the new value presented on the AHB databus.
Reading this register returns the current register content.

30.7.3 HASH start register (HASH_STR)

Address offset: 0x08

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by writing the DCAL bit to 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.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							rw				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

Writing this bit to 1 starts the message padding, using the previously written value of NBLW[4:0], and starts the calculation of the final message digest with all data words written to the input FIFO since the INIT bit was last written to 1.
Reading this bit returns 0.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written in HASH_DIN register, the hash processor takes only the valid bits specified as below, after internal data swapping:
0x00: All 32 bits of the last data written are valid message bits that is M[31:0]
0x01: Only one bit of the last data written (after swapping) is valid that is M[0]
0x02: Only two bits of the last data written (after swapping) are valid that is M[1:0]
0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]
...
0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]
The above mechanism is valid only if DCAL = 0. If NBLW[4:0] bitfield is written while DCAL is set to 1, the NBLW[4:0] bitfield remains unchanged. In other words it is not possible to configure NBLW[4:0] and set DCAL at the same time.
Reading NBLW[4:0] bitfield returns the last value written to NBLW[4:0].

30.7.4 HASH digest registers

These registers contain the message digest result named as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result
- HASH_HR0, HASH_HR1, HASH_HR2 and HASH_HR3 registers return A, B, C and D (respectively), as defined by MD5.
- HASH_HR0 to HASH_HR6 registers return the SHA-224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA-256 digest result.

In all cases, the digest most significant bit is stored in HASH_H0[31] and unused HASH_HRx registers are read as zeros.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation is stalled until the hash calculation has completed.

Note: When starting a digest computation for a new message (by writing the INIT bit to 1), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses.

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x0C + 0x04 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

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

Bits 31:0 **Hx[31:0]**: Hash data x

Refer to *Section 30.7.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x04 * x, (x = 0 to 4)

Reset value: 0x0000 0000

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

Bits 31:0 **Hx[31:0]**: Hash data x
Refer to *Section 30.7.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x04 * x, (x = 5 to 7)

Reset value: 0x0000 0000

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

Bits 31:0 **Hx[31:0]**: Hash data x
Refer to *Section 30.7.4: HASH digest registers* introduction.

30.7.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

30.7.6 HASH status register (HASH_SR)

Address offset: 0x24

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
												r	r	rc_w0	rc_w0

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.
0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by writing the INIT bit to 1 in the HASH_CR register.
0: No digest available in the HASH_HRx registers (zeros are returned)
1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.
0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.
When DINIS=0, HASH_CSRx registers reads as zero.

30.7.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task, and when the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + x * 0x4, (x = 0 to 53)

Reset value: 0x0000 0002 (HASH_CSR0)

Reset value: 0x0000 0000 (HASH_CSR1 to 53)

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

Bits 31:0 **CSx[31:0]**: Context swap x

Refer to *Section 30.7.7: HASH context swap registers* introduction.

30.7.8 HASH register map

Table 230 gives the summary HASH register map and reset values.

Table 230. HASH 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	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[1]	Res.	LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Res.	Res.
	Reset value														0		0			0	0	0	0	0	0	0	0	0	0	0	0		
0x04	HASH_DIN	DATAIN[31:16]																															
	Reset value	0	0	0	0	0	0	0	0	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	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x0C	HASH_HRA0	H0[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	HASH_HRA1	H1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	HASH_HRA2	H2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	HASH_HRA3	H3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	HASH_HRA4	H4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																														0	0
0x24	HASH_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	DMAS	DCIS	DINIS
	Reset value																													0	0	0	1
0x28 to 0xF4	Reserved	Res.																															
0x0F8	HASH_CSR0	CS0[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	1	0

Table 230. HASH 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
0x0F8 + 0x4 * x, (x = 1 to 53) Last address: 0x1CC	HASH_CSRx	CSx[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
...																																	
0x1D0 to 0x30C	Reserved	Res.																															
0x310	HASH_HR0	H0[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
0x314	HASH_HR1	H1[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
0x318	HASH_HR2	H2[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
0x31C	HASH_HR3	H3[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
0x320	HASH_HR4	H4[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
0x324	HASH_HR5	H5[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
0x328	HASH_HR6	H6[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
0x32C	HASH_HR7	H7[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 2.3 on page 87* for the register boundary addresses.

31 On-the-fly decryption engine (OTFDEC)

31.1 Introduction

OTFDEC allows on-the-fly decryption of the AHB traffic based on the read request address information. Four independent and non-overlapping encrypted regions can be defined in OTFDEC.

OTFDEC uses AES-128 in counter mode to achieve the lowest possible latency. As a consequence, each time the content of one encrypted region is changed, the entire region must be re-encrypted with a different cryptographic context (key or initialization vector). This constraint makes OTFDEC suitable to decrypt read-only data or code, stored in external NOR Flash.

Note: When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to access the Flash memory using the Memory-mapped mode of the Flash memory controller.

When security is enabled in the product, OTFDEC can be programmed only by a secure host.

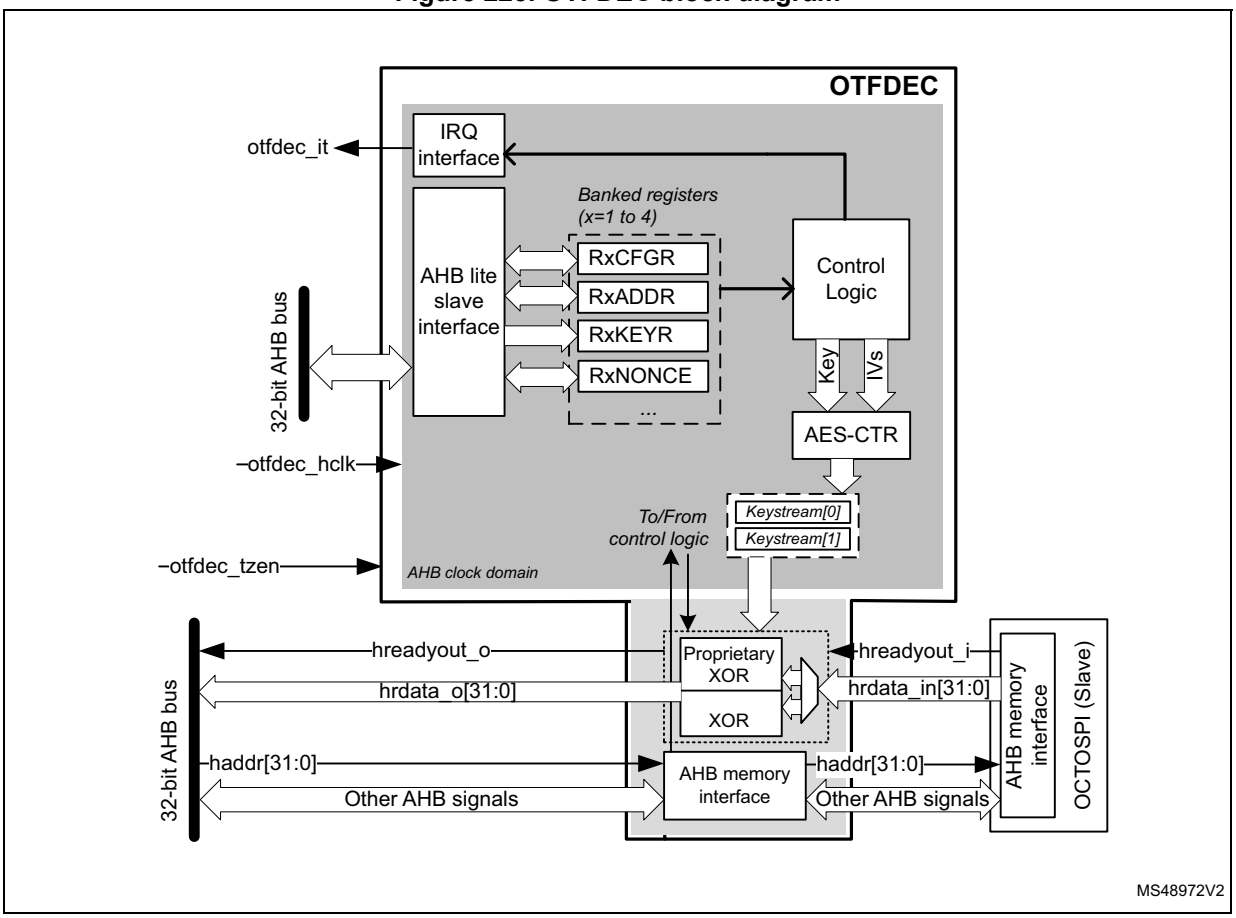
31.2 OTFDEC main features

- On-the-fly 128-bit decryption during the OCTOSPI Memory-mapped read operations (single or multiple).
  - Use of AES in counter (CTR) mode, with two 128-bit keystream buffers
  - Support for any read size
  - Physical address of the reads used for the encryption/decryption
- Up to four independent encrypted regions
  - Granularity of the region definition: 4096 bytes
  - Region configuration write-locking mechanism
  - Each region has its own 128-bit key, two bytes firmware version, and eight bytes application-defined nonce. At least one of those must be changed each time an encryption is performed by the application.
- Encryption keys confidentiality and integrity protection
  - Write-only registers, with software locking mechanism
  - Availability of 8-bit CRC as public key information
- Support for OCTOSPI pre-fetching mechanism
- Possibility to select an enhanced encryption mode to add a proprietary layer of protection on top of AES stream cipher (execute only)
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- Secure only programming if TrustZone security is enabled in the product
- Encryption mode

31.3 OTFDEC functional description

31.3.1 OTFDEC block diagram

Figure 223. OTFDEC block diagram



31.3.2 OTFDEC internal signals

Table 231 describes a list of useful to know internal signals available at OTFDEC level, not at the product level (on pads).

Table 231. OTFDEC internal input/output signals

Signal name	Signal type	Description
otfdec_hclk	Digital input	AHB bus clock
otfdec_it	Digital output	OTFDEC global interrupt request
otfdec_tzen	Digital input	OTFDEC TrustZone enable, controlling TrustZone features of the peripheral (TZEN)

The TZEN option bit in FLASH is used to activate TrustZone in the device.

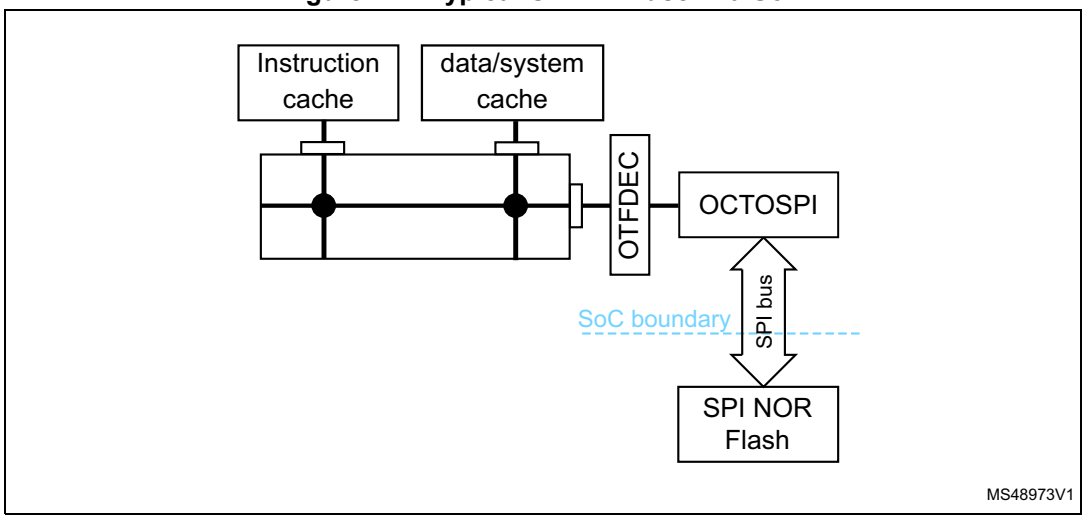
- TZEN = 1: TrustZone security is enabled in the product.
- TZEN = 0: TrustZone security is disabled in the product.

31.3.3 OTFDEC on-the-fly decryption

Introduction

Typical usage for OTFDEC is shown on *Figure 224*.

Figure 224. Typical OTFDEC use in a SoC



Original purpose of OTFDEC is to protect the confidentiality of read-only firmware libraries stored in external SPI NOR Flash devices.

A special locking scheme is available in OTFDEC in order to protect the integrity of the decryption keys and also to protect the other configurations against software denial of services attacks. OTFDEC is only writeable by TrustZone CPU, when TrustZone security is activated.

When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to read the Flash memory using the Memory-mapped mode of the Flash controller.

On top of decrypting on-the-fly, OTFDEC can also encrypt 32-bit word at a time (see *Section 31.5.3: Encrypting for OTFDEC* for more details).

OTFDEC architecture

OTFDEC analyzes all AHB read transfers on the associated AHB bus. If the read request is within one of the four regions programmed in OTFDEC, the control logic triggers a keystream computation based on AES algorithm in counter mode. This keystream is then used to decrypt on-the-fly the data present in the read transfer from the OCTOSPI AHB master, tying low the HREADYOUT signal of this master while the keystream information is being computed (this takes up to 11 cycles). Any accesses outside the enabled OTFDEC regions belong to a non-encrypted region.

Each OTFDEC region is programmed through OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR, OTFDEC_RxNONCER and

OTFDEC_RxKEYR registers, where x = 1 to 4. In OTFDEC_RxCFGR, the MODE bits define the OTFDEC operating mode (standard or enhanced encryption).

Granularity for the region determination is 4096 bytes.

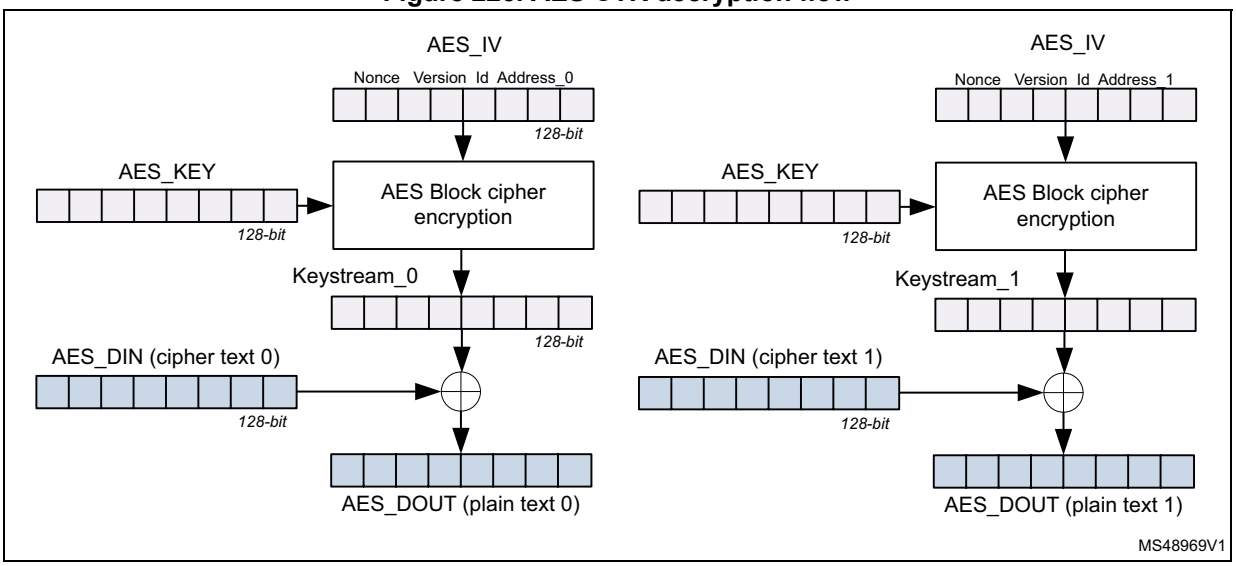
Note: Although OTFDEC does not prevent region overlapping, it is not a valid programming and it must be avoided by application software.

OTFDEC can decrypt incremental or wrap bursts only if they do not cross the 4096-byte aligned address boundaries.

31.3.4 AES in counter mode decryption

Figure 225 shows how OTFDEC uses industry standard Advanced Encryption Standard (AES) algorithm in counter chaining mode. This mode is specified by NIST in *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Figure 225. AES CTR decryption flow



Every 128-bit data block, a special keystream information is computed using AES block cipher, as defined below:

- initialization vector AES_IV[127:0] = RxNONCER1[31:0] || RxNONCER0[31:0] || 0b0000 0000 0000 0000 || RxCFGR[31:16] || 0b00 || (x-1) || ReadAddress[31:4]
- key material AES_KEY[127:0] = R*x*KEYR3[31:0] || R*x*KEYR2[31:0] || R*x*KEYR1[31:0] || R*x*KEYR0[31:0]

Note: Above x is the RegionID of the selected encrypted region (x=1 to 4).
ReadAddress is the AHB address of the encrypted data block, modulo 128-bit.

Resulting 128-bit keystream is XORed with 128-bit cipher text data to produce the 128-bit clear text data.

- AES_DIN and AES_DOUT data blocks are constructed following the rule below (“|” represents a binary concatenation):
  AES_Dx[127:0]= AHB_word(@ | 0xC)[31:0] | AHB_word(@ | 0x8)[31:0] | AHB_word(@ | 0x4)[31:0] || AHB_word(@ | 0x0)[31:0], where @ is the hexadecimal address used to compute the keystream (ReadAddress[31:4] above).

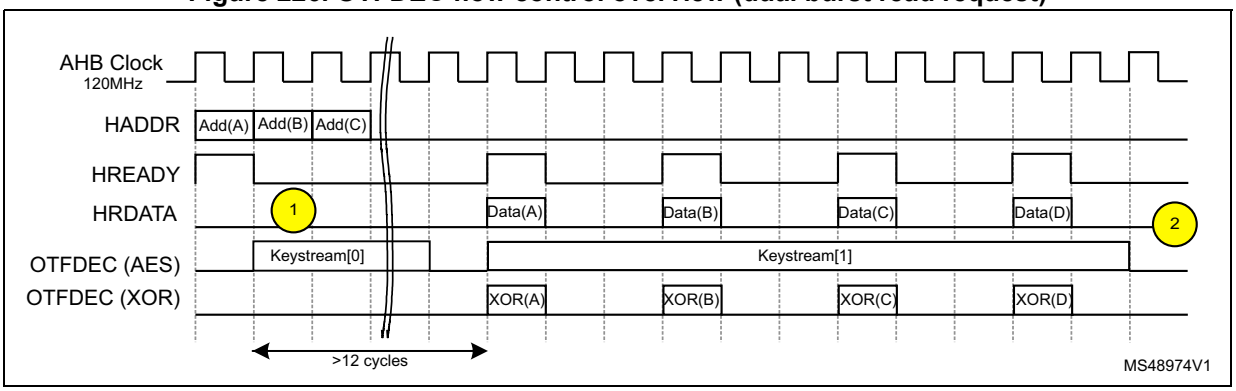
When the read request is not within an encrypted region, or the decryption is not enabled in this region, the AHB data is not changed.

Note: When the application sets the MODE bitfield to 11 in OTFDEC_RxCFGR, an additional layer of protection is added on top of the AES stream cipher. This enhanced encryption mode can only be used with instructions (execute-only region).

31.3.5 Flow control management

Figure 226 shows how OTFDEC manages one INCR4 AHB burst that corresponds to one 128-bit AES data block.

Figure 226. OTFDEC flow control overview (dual burst read request)



with the following notes:

1. OTFDEC enforces HREADY signal from the AHB master low as it is not ready to decrypt data (keystream computation).
2. Thanks to the keystream buffer, OTFDEC can be ready to process a new batch of data within 12 cycles in this configuration (120 MHz AHB clock, 104 MHz SPI bus delivering 2 bytes per SPI clock).

31.3.6 OTFDEC error management

OTFDEC automatically manages errors defined as below:

- Illegal read to OTFDEC_RxKEYR registers
- Illegal write to OTFDEC_RxKEYR registers while CONFIGLOCK or KEYLOCK = 1 in OTFDEC_RxCFGR, while the access is secure. If the security is disabled in the product, the same error occurs when the access is non-secure.
- Illegal write to OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR or OTFDEC_RxNONCER registers while CONFIGLOCK = 1 in OTFDEC_RxCFGR (x = 1 to 4), while the access is secure. If the security is disabled in the product the same error occurs when the access is non-secure.
- Illegal read to an execute-only region (MODE[1:0] = 11). Such illegal request returns 0x0, without bus error.
- Execution request to a region while encryption is enabled (ENC = 1). The request returns 0x0, without bus error.
- Key error: read request to an encrypted region while its key registers are null or not properly initialized (KEYCRC=0x0). Source of the error can be an incorrect key loading sequence (see KEYCRC in OTFDEC_RxCFGR) or it can be an abort event (tamper

detection, unauthorized debug connection, untrusted boot, RDP level regression). Such read request returns 0x0, without bus error.

- Write to any registers while the access is non-secure, if TrustZone security is enabled in the product.

  This last error is managed and cleared through TrustZone interrupt controller, as described in the GTZC section of the product reference manual.

For these errors (except the last one), an interrupt can be generated if the SEIE, XONEIE or KEIE bit is set in OTFDEC_IER register (see *Section 31.4*).

Note: After a key error, OTFDEC keys must be properly initialized again, and a reset of OTFDEC may be needed if registers are locked.

31.4 OTFDEC interrupts

There are three independent maskable interrupt sources generated by the OTFDEC, signaling following security events:

- Illegal read or write access to keys (SEIF flag), see *Section 31.3.6*
- Illegal write to a region configuration while CONFIGLOCK = 1 (SEIF flag), see *Section 31.3.6*
- Read access to an execute-only region (MODE[1:0] = 11), triggering the XONEIF flag
- Executing while encryption is enabled (XONEIF flag)
- Key error (encrypted regions read as zero) triggering the KEIF flag, see *Section 31.3.6*.

Interrupt sources are connected to the same global interrupt request signal.

OTFDEC interrupt sources can be enabled/disabled by setting the corresponding SEIE, XONEIE or KEIE bit in OTFDEC_IER, as described in *Table 232*. Status of the interrupt event is found in OTFDEC_ISR, and this event can be cleared using OTFDEC_ICR.

Table 232. OTFDEC interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method
OTFDEC	Security error	SEIF	SEIE	Set SEIF in OTFDEC_ICR
	Execute-only Execute while encryption	XONEIF	XONEIE	Set XONEIF in OTFDEC_ICR
	Key error	KEIF	KEIE	Set KEIF in OTFDEC_ICR

1. The event flags are found in the OTFDEC_ISR register.

31.5 OTFDEC application information

31.5.1 OTFDEC initialization process

Introduction

One key aspect of OTFDEC is the trusted initialization of its registers, as it involves secret keys. Two trusted initialization schemes are recommended here below.

Note: Those sequences are for production code, as during firmware development, it is not always recommended to lock the key or the region configuration.

Writes to configuration registers are effective when the configuration locks allow it, even if the region is enabled.

Initialization scheme 1: one key for all regions

In this scheme, one entity owns the secret key used to decrypt the four protected regions. The recommended OTFDEC configuration sequence is described below:

1. For *x* = 1 to 4, write the correct MODE[1:0] value in OTFDEC_RxCFGR.
2. For *x* = 1 to 4, program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
3. For *x* = 1 to 4, check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers in this region *x* are no more writable).
4. To do to decrypt a region x (task that does not necessarily have to be performed by the entity that owns the decryption keys):
   a) Verify if the key CRC corresponds to the encrypted binary stored in the region.
   b) Fill the detailed information corresponding to this binary (nonce, start address, end address, version number).
   c) Enable decryption of this region using REG_EN.
   d) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence, step 1 above must be done before step 2, and MODE bits must not be modified after step 2.

Initialization scheme 2: one key per region

In this scheme, one entity can own the secret used to decrypt one (or more) protected region. The recommended OTFDEC configuration sequence is described below:

1. To do to decrypt a region x (this task **must** be performed by the entity that owns the corresponding key):
   a) Write the correct MODE[1:0] value in OTFDEC_R*x*CFGR.
   b) Program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
   c) Check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers are no more writable).
   d) Fill the detailed information corresponding to the protected firmware (nonce, start address, end address, version number).
   e) Enable decryption of this region using REG_EN.
   f) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence step a) above must be done before step b), and MODE bits must not be modified after step b).

31.5.2 OTFDEC and power management

Each time OTFDEC is reset, the correct key loading sequence described in *Section 31.5.1* must be performed (in this case KEYCRC = 0 in OTFDEC_RxCFGR).

It is recommended for application software to verify this point each time OTFDEC is reset by hardware.

31.5.3 Encrypting for OTFDEC

Code and data standard encryption

OTFDEC uses standard AES in counter mode when processing a binary stored in a protected region with MODE[1:0] = 10. When this mode is selected, any AES compatible hardware accelerator or library can be used to encrypt those protected libraries. OTFDEC can be used as well, as described in enhanced encryption section below (with MODE[1:0] = 10).

Definition and endianness of the AES inputs and outputs are defined in *Section 31.3.4: AES in counter mode decryption*.

Enhanced encryption with OTFDEC

OTFDEC uses a proprietary layer of protection on top of the standard AES in counter mode when processing a code stored in a protected region with MODE[1:0] = 11.

Enhanced encryption mode can be used to increase the robustness against tampering.

Recommended sequence to encrypt using OTFDEC is described below:

1. The application in charge of the encryption sets the ENC bit in OTFDEC_CR. This application must run in TrustZone secure mode when TrustZone security is enabled in the product. If PRIV bit is set in OTFDEC_PRIVCFGR, this application must be privileged.
2. Encryption application initializes OTFDEC as described in *Section 31.5.1: OTFDEC initialization process*. OCTOSPI must also be properly clocked, so that OTFDEC is fully functional in encryption mode. This step can also be done before step 1.
3. Encryption application writes 32-bit of clear-text data at the expected protected address, then reads it back encrypted at the same address to store it in RAM. Note that this data stays inside the device, as it is intercepted by OTFDEC in encryption mode.
4. Encryption application goes back to previous step (changing the address) until the whole binary is processed.
5. Encryption application clears the ENC bit in OTFDEC_CR. Another application can then take the encrypted binary and flash it to the correct address in external Flash.

There are few important notes about this procedure:

- Encryption granularity is 32-bit (single 32-bit access is mandatory).
- While ENC bit is set, reads to non-encrypted regions return normal data (such as no encryption nor decryption). While in encryption mode, no access to OCTOSPI (including registers) must be done. This is because the OTFDEC cuts the communication with OCTOPSI while ENC bit is set.
- OTFDEC does not support execution while ENC = 1 (only encrypted data reads). Upon illegal execution detection a XONEIF flag is raised and zero is returned.

31.5.4 OTFDEC key CRC source code

Below is the CRC source code that can be used to compare with the result of the computation provided by OTFDEC in KEYCRC bitfield after loading the keys in OTFDEC_RxKEYR registers.

```
uint8_t getCRC(uint32_t * keyin)
{
  const uint8_t CRC7_POLY = 0x7;
  const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0};
  uint8_t  i, j, k, crc = 0x0;
  uint32_t  keyval;

  for (j = 0; j < 4; j++)
  {
    keyval = *(keyin+j);
    if (j == 0)
    {
      keyval ^= key_strobe[0];
    }
    else
    {
      keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8)
| crc;
    }

    for (i = 0, crc = 0; i < 32; i++)
    {
      k = (((crc >> 7) ^ (keyval >> (31-i))&0xF)) & 1;
      crc <<= 1;
      if (k)
      {
        crc ^= CRC7_POLY;
       }
    }
    crc^=0x55;
  }
  return crc;
}
```

31.6 OTFDEC registers

31.6.1 OTFDEC control register (OTFDEC_CR)

Address offset: 0x0

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENC**: Encryption mode bit

When this bit is set, OTFDEC is used in encryption mode, during which application can write clear text data then read back encrypted data. When this bit is cleared (default), OTFDEC is used in decryption mode, during which application only read back decrypted data. For both modes, cryptographic context (keys, nonces, firmware versions) must be properly initialized. When this bit is set, only data accesses are allowed (zeros are returned otherwise, and XONEIF is set). When MODE = 11, enhanced encryption mode is automatically selected.

0: OTFDEC working in decryption mode

1: OTFDEC working in encryption mode

Note: When ENC bit is set, no access to OCTOSPI must be done (registers and Memory-mapped region).

31.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR)

Address offset: 0x10

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PRIV:** Privileged access protection.

0: No additional protection is added on OTFDEC register accesses.

1: An additional protection is added when accessing all registers except OTFDEC_PRIVCFGR:

– Unprivileged read accesses to registers return zeros

– Unprivileged write accesses to registers are ignored.

Note: This bit can only be written in privileged mode. There is no limitations on reads.

31.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR)

Address offset: 0x20 + 0x30 * (x -1) (x = 1 to 4)

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

Writes are ignored if CONFIGLOCK bit is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REGx_VERSION[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
KEYCRC[7:0]								Res.	Res.	MODE[1:0]		Res.	KEYLOCK	CONFIGLOCK	REG_EN
r	r	r	r	r	r	r	r			rw	rw		rs	rs	rw

Bits 31:16 **REGx_VERSION[15:0]**: region firmware version

This 16-bit bitfield must be correctly initialized before the region corresponding REG_EN bit is set in OTFDEC_RxCFGR.

Bits 15:8 **KEYCRC[7:0]**: region key 8-bit CRC

When KEYLOCK = 0, KEYCRC bitfield is automatically computed by hardware while loading the key of this region in this exact sequence: KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). A new computation starts as soon as a new valid sequence is initiated, and KEYCRC is read as zero until a valid sequence is completed.

When KEYLOCK = 1, KEYCRC remains unchanged until the next reset.

CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm $X^8 + X^2 + X + 1$ (according the convention). Source code is available in *Section 31.5.4*.

This field is read only.

Note: CRC information is updated only after the last bit of the key has been written.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MODE[1:0]**: operating mode
This bitfield selects the OTFDEC operating mode for this region:
10: All read accesses are decrypted (instruction or data).
11: Enhanced encryption mode is activated, and only instruction accesses are decrypted
Others: Reserved
When MODE ≠ 11, the standard AES encryption mode is activated.
When either of the MODE bits are changed, the region key and associated CRC are zeroed.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **KEYLOCK**: region key lock
0: Writes to this region KEYRx registers are allowed.
1: Writes to this region KEYRx registers are ignored until next OTFDEC reset. KEYCRC bitfield is locked.
Note: This bit is set once: if this bit is set, it can only be reset to 0 if the OTFDEC is reset.

Bit 1 **CONFIGLOCK**: region config lock
0: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are allowed.
1: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are ignored until next OTFDEC reset.
Note: This bit is set once. If this bit is set, it can only be reset to 0 if OTFDEC is reset. Setting this bit forces KEYLOCK bit to 1.

Bit 0 **REG_EN**: region on-the-fly decryption enable
0: On-the-fly decryption is disabled for this region.
1: On-the-fly decryption is enabled for this region. Data are XORed with the corresponding keystream.
Note: Garbage is decrypted if region context (version, key, nonce) is not valid when this bit is set.

31.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR)

Address offset: 0x24 + 0x30 * (x -1) (x = 1 to 4)

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REGx_START_ADDR[31:0]**: Region AHB start address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR register is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits and the 12 LSB bits return zeros .

31.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR)

Address offset: 0x28 + 0x30 * (x -1) (x= 1 to 4)

Reset value: 0x0000 0FFF

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REGx_END_ADDR[31:0]**: Region AHB end address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set, and OTFDEC_RxENDADDR must be strictly greater than OTFDEC_RxSTARTADDR to be valid.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in OTFDEC_RxCFGR is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits return zeros and the 12 LSB bits return ones.

31.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0)

Address offset: 0x2C + 0x30 * (x -1) (x = 1 to 4)

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REGx_NONCE[31:0]**: Region nonce, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Writing is discarded in this register if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR is set.

31.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1)

Address offset: 0x30 + 0x30 * (x -1) (x = 1 to 4)

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REGx_NONCE[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
REGx_NONCE[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REGx_NONCE[63:32]**: Region nonce, bits [63:32]

Refer to the OTFDEC_RxNONCER0 register for description of the NONCE[63:0] bitfield.

31.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0)

Address offset: 0x34 + 0x30 * (x -1) (x = 1 to 4)

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REGx_KEY[31:0]**: Region key, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Reading this register returns a zero value. Writing to this register is discarded if performed while the region CONFIGLOCK or KEYLOCK bit is set in the OTFDEC_RxCFGR.

Note: When application successfully changes MODE bits in OTFDEC*_RxCFGR and* OTFDEC*_RxKEYR, and associated KEYCRC are erased.*

31.6.9 OTFDEC region x key register 1 (OTFDEC_RxKEYR1)

Address offset: 0x38 + 0x30 * (x -1) (x = 1 to 4)

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REGx_KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REGx_KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REGx_KEY[63:32]**: Region key, bits [63:32]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

31.6.10 OTFDEC region x key register 2 (OTFDEC_RxKEYR2)

Address offset: 0x3C + 0x30 * (x -1) (x = 1 to 4)

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REGx_KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REGx_KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REGx_KEY[95:64]**: Region key, bits [95:64]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

31.6.11 OTFDEC region x key register 3 (OTFDEC_RxKEYR3)

Address offset: 0x40 + 0x30 * (x -1) (x = 1 to 4)

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REGx_KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REGx_KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REGx_KEY[127:96]**: Region key, bits [127:96]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

31.6.12 OTFDEC interrupt status register (OTFDEC_ISR)

Address offset: 0x300

Reset value: 0x0000 0000

Unprivileged reads return zero if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag status

This bit is set by hardware and read only by application. The bit is set when a read access occurs on an encrypted region, while its key registers is null or not properly initialized (KEYCRC = 0x0).

This bit is cleared when the application sets in OTFDEC_ICR the corresponding bit to 1.

0: OTFDEC operates properly.

1: Read access detected on an enabled encrypted region with its key registers null or not properly initialized (KEYCRC = 0x0). OTFDEC returns a zeroed value for the read, and an optional interrupt is generated if bit KEIE is set to 1 in OTFDEC_IER.

After KEIF is set any subsequent read to the region with bad key registers returns a zeroed value. This state remains until those key registers are properly initialized (KEYCRC not zero).

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when a read access and not an instruction fetch is detected on any encrypted region with MODE bits set to 11. Lastly, XONEIF is also set when an execute access is detected while encryption mode is enabled.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No execute-only error status. No interrupt pending.

1: Read access detected on one region with MODE bits set to 11 or execute access detected while ENC = 1. OTFDEC returns a zeroed value for the illegal access, and an optional interrupt is generated if bit XONEIE is set to 1 in OTFDEC_IER.

Bit 0 **SEIF:** Security error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when at least one security error has been detected.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No security error status. No interrupt pending.

1: Security error flag status, with interrupt pending. Actual interrupt generation is dependent on OTFDEC_IER corresponding bit SEIE.

31.6.13 OTFDEC interrupt clear register (OTFDEC_ICR)

Address offset: 0x304

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag clear
This bit is written by application, and always read as 0.
0: KEIF flag status is not affected.
1: KEIF flag status is cleared in OTFDEC_ISR.

Note: Clearing KEIF does not solve the source of the problem (bad key registers). To be able to access again any encrypted region, OTFDEC key registers must be properly initialized again.

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag clear
This bit is written by application, and always read as 0.
0: XONEIF flag status is not affected.
1: XONEIF flag status is cleared in OTFDEC_ISR.

Bit 0 **SEIF:** Security error interrupt flag clear
This bit is written by application, and always read as 0.
0: SEIF flag status is not affected.
1: SEIF flag status is cleared in OTFDEC_ISR.

31.6.14 OTFDEC interrupt enable register (OTFDEC_IER)

Address offset: 0x308

Reset value: 0x0000 0000

Non-TrustZone AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIE:** Key error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when KEIF flag status is set.

0: Interrupt generation on key error flag KEIF is disabled (masked).

1: Interrupt generation on key error flag KEIF is enabled (not masked).

Bit 1 **XONEIE:** Execute-only execute-never error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when XONEIF flag status is set.

0: Interrupt generation on execute-only error XONEIF is disabled (masked).

1: Interrupt generation on execute-only error XONEIF is enabled (not masked).

Bit 0 **SEIE:** Security error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when SEIF flag status is set.

0: Interrupt generation on security error SEIF is disabled (masked).

1: Interrupt generation on security error SEIF is enabled (not masked).

31.6.15 OTFDEC register map

Table 233. OTFDEC 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	OTFDEC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC
	Reset value																																0
0x10	OTFDEC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV
	Reset value																																0
0x14-0x1C	Reserved	Reserved																															
0x20	OTFDEC_ R1CFGR1	REG1_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_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	0	0	0	0
0x24	OTFDEC_ R1STARTADDR	REG1_START_ADDR[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
0x28	OTFDEC_ R1ENDADDR	REG1_END_ADDR[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
0x2C	OTFDEC_ R1NONCER0	REG1_NONCE[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
0x30	OTFDEC_ R1NONCER1	REG1_NONCE[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
0x34	OTFDEC_ R1KEYR0	REG1_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
0x38	OTFDEC_ R1KEYR1	REG1_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
0x3C	OTFDEC_ R1KEYR2	REG1_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
0x40	OTFDEC_ R1KEYR3	REG1_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
0x4C	Reserved	Reserved																															
0x50	OTFDEC_ R2CFGR	REG2_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_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	0	0	0	0
0x54	OTFDEC_ R2STARTADDR	REG2_START_ADDR[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
0x58	OTFDEC_ R2ENDADDR	REG2_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 233. OTFDEC 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
0x5C	OTFDEC_ R2NONCER0	REG2_NONCE[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
0x60	OTFDEC_ R2NONCER1	REG2_NONCE[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
0x64	OTFDEC_ R2KEYR0	REG2_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
0x68	OTFDEC_ R2KEYR1	REG2_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
0x6C	OTFDEC_ R2KEYR2	REG2_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
0x70	OTFDEC_ R2KEYR3	REG2_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
0x7C	Reserved	Reserved																															
0x80	OTFDEC_ R3CFGR	REG3_VERSION[15:0]																KEYCRC[7:0]								Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_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	0	0	0	0
0x84	OTFDEC_ R3STARTADDR	REG3_START_ADDR[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
0x88	OTFDEC_ R3ENDADDR	REG3_END_ADDR[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
0x8C	OTFDEC_ R3NONCER0	REG3_NONCE[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
0x90	OTFDEC_ R3NONCER1	REG3_NONCE[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
0x94	OTFDEC_ R3KEYR0	REG3_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
0x98	OTFDEC_ R3KEYR1	REG3_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
0x9C	OTFDEC_ R3KEYR2	REG3_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
0xA0	OTFDEC_ R3KEYR3	REG3_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
0xAC	Reserved	Reserved																															

Table 233. OTFDEC 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
0xB0	OTFDEC_ R4CFGR	REG4_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_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	0	0	0	0
0xB4	OTFDEC_ R4STARTADDR	REG4_START_ADDR[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
0xB8	OTFDEC_ R4ENDADDR	REG4_END_ADDR[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
0xBC	OTFDEC_ R4NONCER0	REG4_NONCE[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
0xC0	OTFDEC_ R4NONCER1	REG4_NONCE[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
0xC4	OTFDEC_ R4KEYR0	REG4_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
0xC8	OTFDEC_ R4KEYR1	REG4_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
0xCC	OTFDEC_ R4KEYR2	REG4_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
0xD0	OTFDEC_ R4KEYR3	REG4_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
0x0D4-0x2FC	Reserved	Reserved																															
0x300	OTFDEC_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.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
	Reset value																														0	0	0
0x304	OTFDEC_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.	Res.	KEIF	XONEIF	SEIF
	Reset value																														0	0	0
0x308	OTFDEC_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.	Res.	Res.	Res.	Res.	KEIE	XONEIE	SEIE
	Reset value																														0	0	0

Refer to *Section 2.3* for the register boundary addresses.

32 Public key accelerator (PKA)

32.1 Introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(*p*) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

All needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

32.2 PKA main features

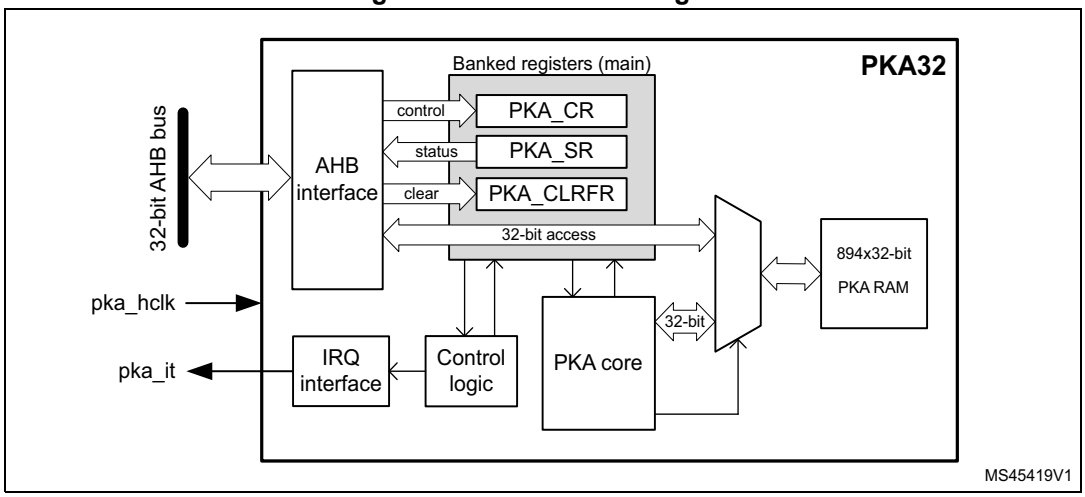
- Acceleration of RSA, DH and ECC over GF(*p*) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA Chinese Remainder Theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check
  - ECDSA signature generation and verification
- Capability to handle operands up to 3136 bits for RSA/DH and 640 bits for ECC.
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication.
- Built-in Montgomery domain inward and outward transformations.
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise, for writes, an AHB bus error is generated, and write accesses are ignored).

32.3 PKA functional description

32.3.1 PKA block diagram

Figure 227 shows the block diagram of the public key accelerator PKA.

Figure 227. PKA block diagram



32.3.2 PKA internal signals

Table 234 lists internal signals available at the IP level, not necessarily available on product bonding pads.

Table 234. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request

32.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. The RAM receives this clock directly, the core is clocked at half the frequency.

When the PKA peripheral reset signal is released PKA RAM is cleared automatically, taking 894 clock cycles. During this time the setting of EN bit in PKA_CR is ignored.

According to the security policy applied to the device PKA RAM can also be reset following a tamper event. Refer to the Tamper detection and response in the System security section (if applicable to this product).

32.3.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 3136 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information is found in *Section 32.5.1: Supported elliptic curves*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 3576 bytes (894 words of 32 bits) called PKA RAM is used for providing initial data to the PKA, and for holding the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 235* and *Table 236*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register.

Table 235. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R2 mod n	*Section 32.4.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 32.4.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 32.4.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 32.4.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 32.4.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x08	001000	Modular inversion A-1 mod n	*Section 32.4.7*
0x0D	001101	Modular reduction A mod n	*Section 32.4.8*
0x09	001001	Arithmetic addition A+B	*Section 32.4.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 32.4.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 32.4.11*
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 32.4.12*
0x07	000111	RSA CRT exponentiation	*Section 32.4.13*

Table 236. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 32.4.14*
0x20	100000	ECC scalar multiplication kP	*Section 32.4.15*
0x22	100010	ECC scalar multiplication kP (fast mode)	
0x24	100100	ECDSA sign	*Section 32.4.16*
0x26	100110	ECDSA verification	*Section 32.4.17*

Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is supplied before starting the operation. Performance improvement is detailed in *Section 32.5.2: Computation times*.

The operations using fast mode are modular exponentiation and scalar multiplication.

32.3.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions. In particular:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

Specifications of the above functions are given in following publications:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

RSA key pair

For following RSA operations a public key and a private key information are defined as below:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information on above representations refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (*n*, *e*) must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer m, with $0 \leq m < n$ and (m, n) being co-primes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext m is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\ (m_1 - m_2) \bmod p$
   d) $m = m_2 + h\ q$
3. Convert the integer message representative m to an encoded message EM.
4. Recover message M= DECODE(EM), where DECODE is a decoding method.

Above operations can be accelerated by PKA using *Modular exponentiation* $A^e \bmod n$ if the private key is d, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

Elliptic curve selection

For following ECC operations curve parameters are defined as below:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 32.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (i.e. n x G = identity element O).

ECDSA message signature generation

ECDSA (Elliptic Curve Digital Signature Algorithm) signature generation function principle is the following: Alice, to sign a message m using her private key integer d_A, follows the steps below.

1. Calculate e = HASH(m), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order n.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1} (z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (r, s).

Steps 4 to 7 are accelerated by PKA using:

- *ECDSA sign* or
- All of the operations below:
  - *ECC Fp scalar multiplication* k x P
  - *Modular reduction* A mod *n*
  - *Modular inversion* A^{-1} mod *n*
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A.

Bob can verify that Q_A is a valid curve point going through the following steps:

1. check that Q_A is not equal to the identity element O
2. check that Q_A is on the agreed curve
3. check that n x Q_A = O.

Then Bob follows the procedure detailed below:

1. verify that *r* and *s* are integer in [1, n-1]
2. calculate *e* = HASH(*m*), where HASH is the agreed cryptographic hash function
3. let z be the L_n leftmost bits of *e*
4. calculate w = s^{-1} mod *n*
5. calculate u_1 = zw mod *n* and u_2 = *rw* mod *n*
6. calculate the curve point (x_1, y_1) = u_1 x G + u_2 x Q_A
7. the signature is valid if *r* = x_1 (mod *n*), it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using *ECDSA verification*.

32.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

Note: When PKA is in the process of clearing its memory EN bit cannot be set.

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 32.4*.

Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set to “1”, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA_SR.

Using precomputed Montgomery parameters (PKA fast mode)

As explained in *Section 32.3.4*, when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, *Section 32.4.5*). This is know as “fast mode”.

To manage Fast Mode usage the recommended procedure is described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in *Section 32.5.1*.
2. Program in PKA_CR register the PKA in *Montgomery parameter computation* mode (MODE=”0x1”) then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set to “1”, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.
4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in *Section 32.4*.

32.3.7 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the Address Error flag (ADDRERRF) is set in the PKA_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM Error Flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return zero, while writes are ignored.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see *Section 32.6* for details).

ADDRERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

32.4 PKA operating modes

32.4.1 Introduction

The various operations supported by PKA are described in the following subsections, clarifying the associated format of the input data and of the results, both stored in the PKA RAM.

The following information applies to all PKA operations.

- PKA core processes 32-bit words
- Supported operand “Size” are:
  - ROS (RSA operand size): data size is (*rsa_size*/32+1) words, with *rsa_size* equal to the chosen modulus length. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 33 words, or 1056 bits.
  - EOS (ECC operand size): data size is (*ecc_size*/32+1) words, with *ecc_size* equal to the chosen prime modulus length. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 7 words, or 224 bits.

Note: Fractional results for above formulas are rounded up to the nearest integer since PKA core processes 32-bit words.

Note: The maximum ROS is 99 words (3136-bit max exponent size), while the maximum EOS is 21 words (640-bit max operand size).

- The column indicated with Storage in the following tables indicates either the Register PKA_CR, or the address offset within the PKA, located in the PKA RAM area (starting from 0x400). To recover the physical address of an operand the application must add to the indicated offset the base address of the PKA.
- About writing parameters (“IN” direction)
  - When elements are written as input in the PKA memory, an additional word with all bits equal to zero must be added. As an example, when ECC P256 is used and when loading an input (represented on 256 bits or 8 words), an additional word is expected by the PKA and it has to be filled with zeros.
  - About endianess, for example to prepare the operation ECC Fp scalar multiplication, when application writes x_p coordinate for an ECC P256 curve (EOS= 9 words) the least significant bit is to be placed in bit 0 at address offset 0x55C; the most significant bit is to be placed in bit 31 of address offset 0x578. Then, as mentioned above, word 0x00000000 should also be written at address offset 0x57C.
  - Unless indicated otherwise all operands in the tables are integers.
- About PKA outputs (“OUT” direction)
  - Unless specified in the tables PKA does not provide an error output when a wrong parameter is written by the application.

Caution: Validity of all input parameters to the PKA must be checked before issuing any PKA operation. Indeed, the PKA assumes that all input parameters are valid and consistent with each other.

32.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod n) used by PKA to convert operands into the Montgomery residue system representation.

Note: This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.

Operation instructions for Montgomery parameter computation are summarized in *Table 237*.

Table 237. Montgomery parameter computation

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x01	PKA_CR	6 bits
	Modulus length	(In bits, 0 ≤ value < 3136bits)	RAM@0x404	32 bits
	Modulus value n	(Odd integer only, n < 2^{3136})	RAM@0xD5C	ROS
OUT	Result: R^2 n	-	RAM@0x594	

32.4.3 Modular addition

Modular addition operation consists in the computation of A + B mod *n*. Operation instructions are summarized in *Table 238*.

Table 238. Modular addition

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0E	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ A < n)	RAM@0x8B4	ROS
	Operand B	(0 ≤ B < n)	RAM@0xA44	
	Modulus value n	(n < 2^{3136})	RAM@0xD5C	
OUT	Result: A+B mod n	(0 ≤ result < n)	RAM@0xBD0	

32.4.4 Modular subtraction

Modular subtraction operation consists in the following computations:

- If A ≥ B result equals A - B mod *n*
- If A < B result equals A + n - B mod *n*

Operation instructions are summarized in *Table 239*.

Table 239. Modular subtraction

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0F	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ A < n)	RAM@0x8B4	ROS
	Operand B	(0 ≤ B < n)	RAM@0xA44	
	Modulus value n	(n < 2^{3136})	RAM@0xD5C	
OUT	Result: A-B mod n	(0 ≤ result < n)	RAM@0xBD0	

32.4.5 Modular and Montgomery multiplication

In order to be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has input and outputs in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication A x B mod *n*

The method to perform above operations are described below. Note that “x” function is this operation, and A, B, C operands are in the natural domain.

1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Let’s assume A is an integer in the natural domain
      Compute *r2modn* using *Montgomery parameter computation*
      Result AR= A x *r2modn* mod *n* is A in the Montgomery domain
   b) Let’s assume BR is an integer in the Montgomery domain
      Result B = BR x 1 mod *n* is B in the natural domain
      Similarly, above value AR computed in a) can be converted into the natural domain by computing A = AR x 1 mod *n*
2. Simple modular multiplication A x B mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain
   c) Compute AB= AR x B mod *n*. Output is in natural domain
3. Multiple modular multiplication A x B x C mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain
   c) Compute BR = B x *r2modn* mod *n*. Output is in the Montgomery domain
   d) Compute ABR= AR x BR mod *n*. Output is in the Montgomery domain
   e) Compute CR = C x *r2modn* mod *n*. Output is in the Montgomery domain
   f) Compute ABCR= ABR x CR mod *n*. Output is in the Montgomery domain
   g) (optional) Repeat the two steps above if more operands need to be multiplied
   h) Compute ABC= ABCR x 1 mod *n* to retrieve the result in natural domain

Operation instructions for Montgomery multiplication are summarized in *Table 240*.

Table 240. Montgomery multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x10	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ A < n)	RAM@0x8B4	ROS
	Operand B	(0 ≤ B < n)	RAM@0xA44	
	Modulus value n	(Odd integer only, n < 2^{3136})	RAM@0xD5C	
OUT	Result: AxB mod n	-	RAM@0xBD0	

32.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of A^e mod *n*.

Operation instructions for modular exponentiation are summarized in *Table 241* (normal mode) and in *Table 242* (fast mode). Fast mode usage is explained in *Section 32.3.6*.

Table 241. Modular exponentiation (normal mode)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	32 bits
	Operand length	(in bits, not null)	RAM@0x404	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xA44	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xBD0	
	Modulus value n	(Odd integer only, $n < 2^{3136}$)	RAM@0xD5C	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x724	

Table 242. Modular exponentiation (fast mode)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	32 bits
	Operand length	(in bits, not null)	RAM@0x404	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xA44	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xBD0	
	Modulus value n	(Odd integer only, $n < 2^{3136}$)	RAM@0xD5C	
IN/OUT	Montgomery param R2 mod n	(mandatory)	RAM@0x594	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x724	

32.4.7 Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse A^{-1} mod *n*. If the modulus *n* is prime, for all values of A (1 ≤ A < *n*) modular inversion output is valid. If the modulus *n* is not prime, A has an inverse only if the largest common divisor between A and *n* is 1.

If the operand A is a divisor of the modulus *n,* the result is a multiple of a factor of *n*.

Operation instructions for modular inversion are summarized in *Table 243*.

Table 243. Modular inversion

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x08	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ A < n)	RAM@0x8B4	ROS
	Modulus value n	(Odd integer only, n < 2^{3136})	RAM@0xA44	
OUT	Result: A^{-1} mod n	0 < result < n	RAM@0xBD0	

32.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of A divided by n. Operation instructions are summarized in *Table 244*.

Table 244. Modular reduction

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0D	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x400	32 bits
	Modulus length	(In bits, 8 < value < 3136)	RAM@0x404	
	Operand A	(0 ≤ A < 2n < 2^{3136})	RAM@0x8B4	ROS
	Modulus value n	(Odd integer only, n < 2^{3136})	RAM@0xA44	
OUT	Result A mod n	(0 < result < n)	RAM@0xBD0	

32.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of A + B. Operation instructions are summarized in *Table 245*.

Table 245. Arithmetic addition

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x09	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ < A < 2^{M})	RAM@0x8B4	ROS
	Operand B	(0 ≤ < B < 2^{M})	RAM@0xA44	
OUT	Result: A+B	(0 ≤ result < 2^{M+1})	RAM@0xBD0	ROS + 1

32.4.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If A ≥ B result equals A - B
- If A < B and M/32 residue is >0 result equals A + $2^{int(M/32)*32+1}$ - B
- If A < B and M/32 residue is 0 result equals A + $2^{int(M/32)*32}$ - B

Operation instructions are summarized in *Table 246*.

Table 246. Arithmetic subtraction

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0A	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	($0 \leq < A < 2^M$)	RAM@0x8B4	ROS
	Operand B	($0 \leq < B < 2^M$)	RAM@0xA44	
OUT	Result: A-B	($0 \leq$ result $< 2^M$)	RAM@0xBD0	

32.4.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in *Table 247*.

Table 247. Arithmetic multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0B	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	($0 \leq < A < 2^M$)	RAM@0x8B4	ROS
	Operand B	($0 \leq < B < 2^M$)	RAM@0xA44	
OUT	Result: AxB	($0 \leq$ result $< 2^M$)	RAM@0xBD0	2xROS

32.4.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:

- If A=B then result=0x0
- If A>B then result=0x1
- If A<B then result=0x2

Operation instructions for arithmetic comparison are summarized in *Table 248*.

Table 248. Arithmetic comparison

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0C	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	($0 \leq < A < 2^M$)	RAM@0x8B4	ROS
	Operand B	($0 \leq < B < 2^M$)	RAM@0xA44	
OUT	Result A=B or A>B or A<B	0x0, 0x1 or 0x02	RAM@0xBD0	32 bits

32.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries like OpenSSL RSA use the following optimization for decryption and signing based on the Chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- d_P= d mod (*p* -1)
- d_Q= d mod (*q* -1) and
- q_{inv}= q^{-1} mod *p*

These values allow the recipient to compute the exponentiation m = A^d (mod pq) more efficiently as follows:

- m_1 = A^{dP} mod *p*
- m_2 = A^{dQ} mod *p*
- h = q_{inv} (m_1 - m_2) mod *p*, with m_1> m_2
- m = m_2 + h*q*

Operation instructions for computing CRT exponentiation A^d mod pq are summarized in *Table 249*.

Table 249. CRT exponentiation

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x404	32 bits
IN	Operand d_P	($0 \le d_P < 2^{M/2}$)	RAM@0x65C	ROS/2
	Operand d_Q	($0 \le d_Q < 2^{M/2}$)	RAM@0xBD0	
	Operand q_{inv}	($0 \le q_{inv} < 2^{M/2}$)	RAM@0x7EC	
	Prime *p*(1)	($0 \le p < 2^{M/2}$)	RAM@0x97C	
	Prime *q*(1)	($0 \le q < 2^{M/2}$)	RAM@0xD5C	
IN	Operand A	($0 \le A < 2^{M/2}$)	RAM@0xEEC	ROS
OUT	Result: A^d mod *pq*	(0 ≤ result < *pq*)	RAM@0x724	

1. Must be different from 2.

32.4.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point P (x, y) satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \bmod p$, where *a* and *b* are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in *Table 250*.

Table 250. Point on elliptic curve Fp check

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x28	PKA_CR	6 bits
	Modulus length	(In bits, not null, 8 < value < 640)	RAM@0x404	32 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x408	
	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x40C	EOS
	Curve coefficient *b*	(|b| < p)	RAM@0x7FC	
	Curve modulus value *p*	(Odd integer prime, 0 < p < 2640)	RAM@0x460	
	Point P coordinate *x*	(x < *p*)	RAM@0x55C	
	Point P coordinate *y*	(y < *p*)	RAM@0x5B0	
OUT	Result: P on curve	0x0: point on curve Not 0x0: point not on curve	RAM@0x400	32 bits

32.4.15 ECC Fp scalar multiplication

This operation consists in the computation of a k x P (x_P, y_P), where P is a point on a curve over prime fields and “x” is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in *Table 251* (normal mode) and *Table 252* (fast mode). Fast mode usage is explained in *Section 32.3.6*.

Table 251. ECC Fp scalar multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x20	PKA_CR	6 bits
IN	Scalar multiplier k length	(In bits, not null, 8 < value < 640)	RAM@0x400	32 bits
	Modulus length	(In bits, not null, 8 < value < 640)	RAM@0x404	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x408	
IN	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x40C	EOS
	Curve modulus value *p*	(Odd integer prime, $0 < p < 2^{640}$)	RAM@0x460	
	Scalar multiplier k	($0 \leq k < 2^{640}$)	RAM@0x508	
	Point P coordinate x_P	(x < *p*)	RAM@0x55C	
	Point P coordinate y_P	(y < *p*)	RAM@0x5B0	
OUT	Result: k x P coordinate *x*	(result < *p*)	RAM@0x55C	
	Result: k x P coordinate *y*	(result < *p*)	RAM@0x5B0	

Table 252. ECC Fp scalar multiplication (Fast Mode)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x22	PKA_CR	6 bits
IN	Scalar multiplier k length	(In bits, not null, 8 < value < 640)	RAM@0x400	32 bits
	Modulus length	(In bits, not null, 8 < value < 640)	RAM@0x404	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x408	
IN	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x40C	EOS
	Curve modulus value *p*	(Odd integer prime, $0 < p < 2^{640}$)	RAM@0x460	
	Scalar multiplier k	($0 \leq k < 2^{640}$)	RAM@0x508	
	Point P coordinate x_P	(x < *p*)	RAM@0x55C	
	Point P coordinate y_P	(y < *p*)	RAM@0x5B0	
IN	Montgomery parameter R^2 mod *p*	(mandatory)	RAM@0x4B4	
OUT	Result: k x P coordinate *x*	(result < *p*)	RAM@0x55C	
	Result: k x P coordinate *y*	(result < *p*)	RAM@0x5B0	

When performing this operation following special cases should be noted:

- For k = 0, this function returns a point at infinity, that is (0, 0) if curve parameter b is nonzero and (0, 1) otherwise. For k different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation should be carried out.
- For k < 0 (i.e. a negative scalar multiplication is required) multiplier absolute value k = |-k| should be provided to the PKA. After the computation completion, the formula -P = (x, -y) can be used to compute the y coordinate of the effective final result (the x coordinate remains the same).

32.4.16 ECDSA sign

ECDSA signing operation (outlined in *Section 32.3.5*) is summarized in *Table 253* (input parameters) and in *Table 254* (output parameters).

The application should check if the output error is equal to zero, if it is different from zero a new k should be generated and the ECDSA sign operation should be repeated.

Table 253. ECDSA sign - Inputs

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x24	PKA_CR	6 bits
	Curve prime order n length	(in bits, not null)	RAM@0x400	32 bits
	Curve modulus p length	(in bits, 8 < value < 640)	RAM@0x404	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x408	
	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x40C	EOS
	Curve modulus value *p*	(Odd integer prime, $0 < p < 2^{640}$)	RAM@0x460	
	Integer k(1)	($0 \leq k < 2^{640}$)	RAM@0x508	
	Curve base point G coordinate x	(x < *p*)	RAM@0x55C	
	Curve base point G coordinate y	(y < *p*)	RAM@0x5B0	
	Hash of message z	(z < 2M)	RAM@0xDE8	
	Private key d	(positive integer)	RAM@0xE3C	
	Curve prime order n	(integer prime)	RAM@0xE94	

1. This integer is usually a cryptographically secure random number, but in some cases k could be deterministically generated.

Table 254. ECDSA sign - Outputs

Parameters with direction		Value (Note)	Storage	Size
OUT	Signature part *r*	(0 < *r* < n)	RAM@0x700	EOS
	Signature part *s*	(0 < *s* < n)	RAM@0x754	
ERROR	Result of signature	– 0x0: no error – 0x1: signature part r is equal to 0 – 0x2: signature part s is equal to 0	RAM@0xEE8	32 bits

Note: If error output is different from zero the content of the PKA memory should be cleared to avoid leaking information about the private key.

Extended ECDSA support

PKA also supports Extended ECDSA signature, for which the inputs and the outputs have the same ECDSA signature (*Table 253* and *Table 254*, respectively), with the addition of the coordinates of the point kG. This extra output is defined in *Table 255*.

Table 255. Extended ECDSA sign (extra outputs)

Parameters with direction		Value (Note)	Storage	Size
OUT	Curve point kG coordinate x_1	($0 \le x_1 < p$)	RAM@0x103C	EOS
	Curve point kG coordinate y_1	($0 \le y_1 < p$)	RAM@0x1090	

32.4.17 ECDSA verification

ECDSA verification operation (outlined in *Section 32.3.5*) is summarized in *Table 256* (input parameters) and *Table 257* (output parameters).

The epplication should check if the output error is equal to zero, if it is different from zero, the signature is not verified.

Table 256. ECDSA verification (inputs)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x26	PKA_CR	6 bits
	Curve prime order n length	(In bits, not null)	RAM@0x404	32 bits
	Curve modulus p length	(In bits, not null, 8 < value < 640)	RAM@0x4B4	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x45C	
	Curve coefficient |a|	(Absolute value, |a| < p)	RAM@0x460	EOS
	Curve modulus value *p*	(Odd integer prime, $0 < p < 2^{640}$)	RAM@0x4B8	
	Curve base point G coordinate x	($x < p$)	RAM@0x5E8	
	Curve base point G coordinate y	($y < p$)	RAM@0x63C	
	Public-key curve point Q coordinate x_Q	($x_Q < p$)	RAM@0xF40	
	Public-key curve point Q coordinate y_Q	($y_Q < p$)	RAM@0xF94	
	Signature part *r*	($0 < r < n$)	RAM@0x1098	
	Signature part *s*	($0 < s < n$)	RAM@0xA44	
	Hash of message *z*	($z < 2^M$)	RAM@0xFE8	
	Curve prime order *n*	(integer prime)	RAM@0xD5C	

Table 257. ECDSA verification (outputs)

Parameters with direction		Value (Note)	Storage	Size
OUT	Result: ECDSA verify	0x0: valid signature Not 0x0: invalid signature	RAM@0x5B0	32 bits

32.5 Example of configurations and processing times

32.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curvescan be described with a short Weierstrass equation $y^2 = x^3 + ax + b \pmod{p}$.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field GF(*p*)
- the (usually prime) integer *n*, the order of the group generated by G, defined below
- the base point of the curve G, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. **a defined as *p*-|*a*|** in the finite field GF(*p*), for example **p-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 258 summarizes the family of curves supported by PKA for ECC operations.

Table 258. Family of supported curves for ECC operations

Curve name	Standard	Reference
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4
P-224		
P-256		
P-384		
P-521		

Table 258. Family of supported curves for ECC operations (continued)

Curve name	Standard	Reference	
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

32.5.2 Computation times

The following tables summarize the PKA computation times, expressed in clock cycles.

Table 259. Modular exponentiation computation times

Exponent length (in bits)	Mode	Modulus length (in bits)		
		1024	2048	3072
3	Normal	304000	814000	1728000
	Fast	46000	164000	356000
17	Normal	326000	896000	1910000
	Fast	68000	246000	534000
$2^{16} + 1$	Normal	416000	1222000	2616000
	Fast	158000	572000	1244000
1024	Normal	11664000	-	-
	Fast	11280000	-	-
	CRT(1)	3546000	-	-
2048	Normal	-	83834000	-
	Fast	-	82046000	-
	CRT(1)	-	23468000	-
3072	Normal	-	-	274954000
	Fast	-	-	273522000
	CRT(1)	-	-	73378000

1. CRT stands for chinese remainder theorem optimization (MODE bitfield = 0x07).

Table 260. ECC scalar multiplication computation times(1)

Mode	Modulus length (in bits)						
	160	192	256	320	384	512	521
Normal	1634000	2500000	4924000	8508000	13642000	28890000	33160000
Fast	1630000	2494000	4916000	8494000	13614000	28842000	33158000

1. These times depend on the number of “1”s included in the scalar parameter.

Table 261. ECDSA signature average computation times(1) (2)

Modulus length (in bits)						
160	192	256	320	384	512	521
1760000	2664000	5249000	9016000	14596000	30618000	35540000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 262. ECDSA verification average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**521**
3500000	5350000	10498000	18126000	29118000	61346000	71588000

Table 263. Point on elliptic curve Fp check average computation times

Modulus length (in bits)					
160	**192**	**256**	**320**	**384**	**512**
10800	14200	20400	31000	49600	82400

Table 264. Montgomery parameters average computation times(1)

Modulus length (in bits)									
160	**192**	**256**	**320**	**384**	**512**	**521**	**1024**	**2048**	**3072**
4518	7846	11848	14902	21682	35012	64000	119536	466146	1104642

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

32.6 PKA interrupts

There are three individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. access to unmapped address (ADDRERRF), see *Section 32.3.7*
2. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 32.3.7*
3. PKA end of operation (PROCENDF)

The three interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 265 gives a summary of the available features.

Table 265. PKA interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
PKA	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit

32.7 PKA registers

32.7.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code
000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
100000: Montgomery parameter computation then ECC scalar multiplication
100010: ECC scalar multiplication only (Montgomery parameter must be loaded first)
100100: ECDSA sign
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
Others: Reserved

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

Bit 1 **START**: start the operation

Writing 1 to this bit starts the operation which is selected by MODE[5:0], using the operands and data already written to the PKA RAM. This bit is always read as 0.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable.

0: Disable PKA

1: Enable PKA

Note: When EN=0 PKA RAM can still be accessed by the application.

32.7.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDR ERRF	RAM ERRF	Res.	PROC ENDF	BUSY
											r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADDRERRF**: Address error flag

0: No Address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA End of Operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: PKA operation is in progress

This bit is set to 1 whenever START bit in the PKA_CR is set. It is automatically cleared when the computation is complete, meaning that PKA RAM can be safely accessed and a new operation can be started.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode the peripheral is busy for a couple of cycles, then it aborts automatically the operation and go back to ready (BUSY bit is set to 0).

Bits 15:0 Reserved, must be kept at reset value.

32.7.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDR ERRFC	RAM ERRFC	Res.	PROC ENDFC	Res.
											w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADDRERRFC**: Clear Address error flag
- 0: No action
- 1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
- 0: No action
- 1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA End of Operation flag
- 0: No action
- 1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

32.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 3576 bytes (max word offset: 0x11F4).

Note: The PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 32.3.3

32.7.5 PKA register map

Table 266. PKA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value												0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0		0	0																
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0		0																

Refer to *Section 2.3 on page 87* for the register boundary addresses.

33 Advanced-control timers (TIM1/TIM8)

33.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) 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.

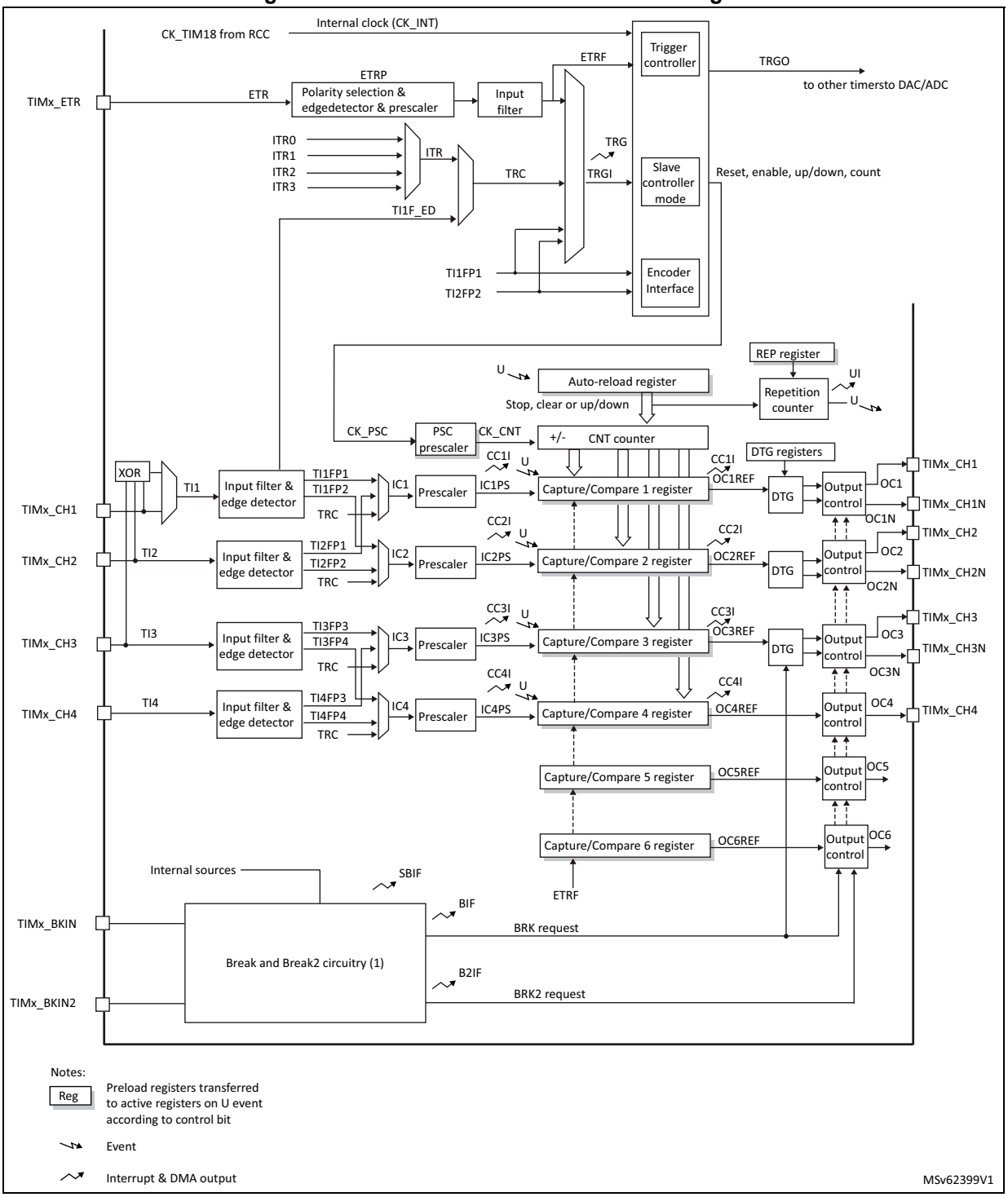
The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 33.3.26: Timer synchronization*.

33.2 TIM1/TIM8 main features

TIM1/TIM8 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 228. Advanced-control timer block diagram



1. See *Figure 272: Break and Break2 circuitry overview* for details

33.3 TIM1/TIM8 functional description

33.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 229 and *Figure 230* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 229. Counter timing diagram with prescaler division change from 1 to 2

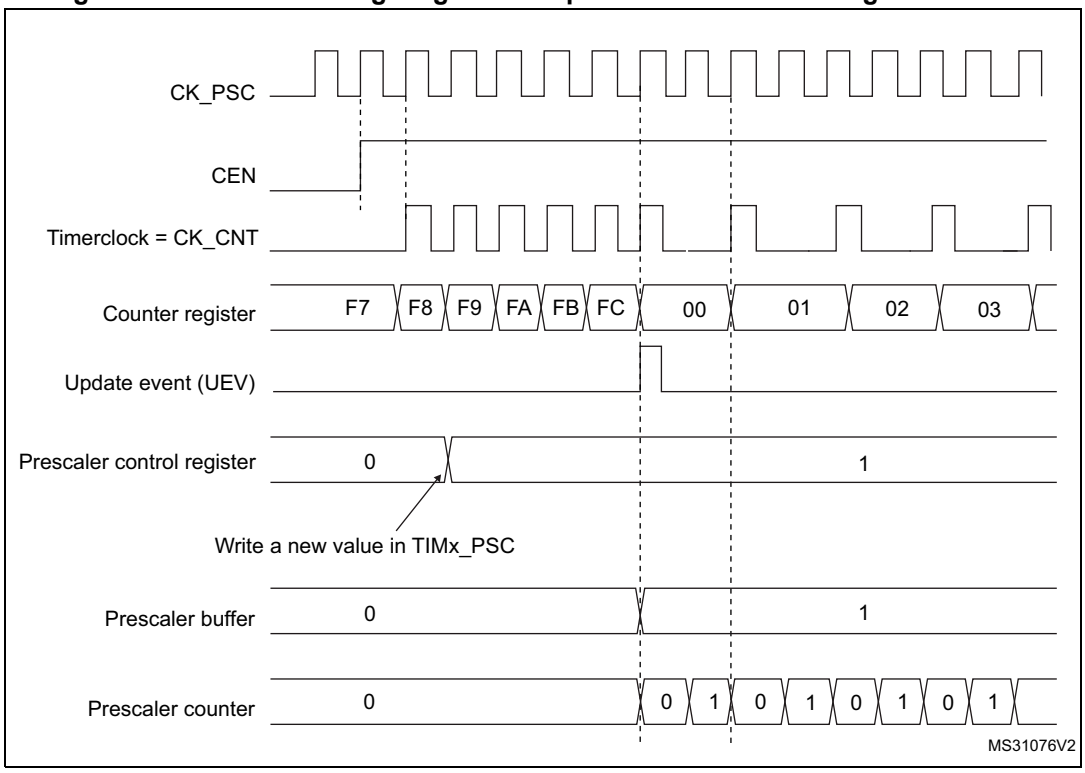
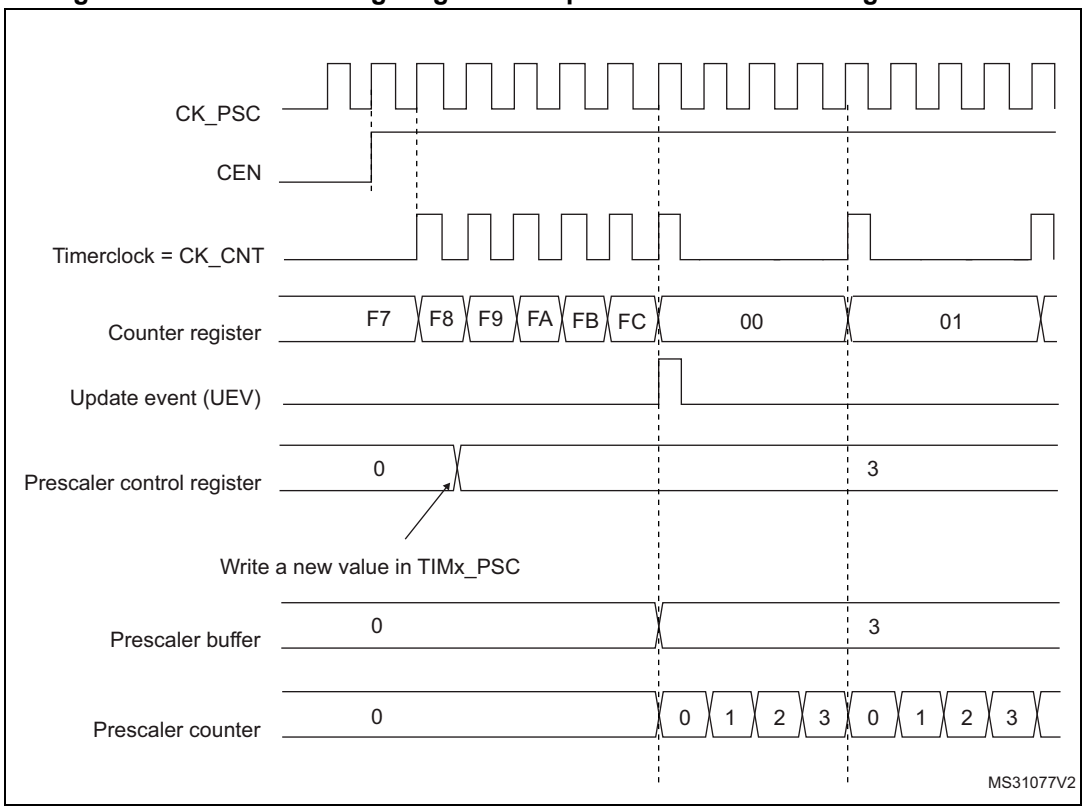


Figure 230. Counter timing diagram with prescaler division change from 1 to 4



33.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 231. Counter timing diagram, internal clock divided by 1

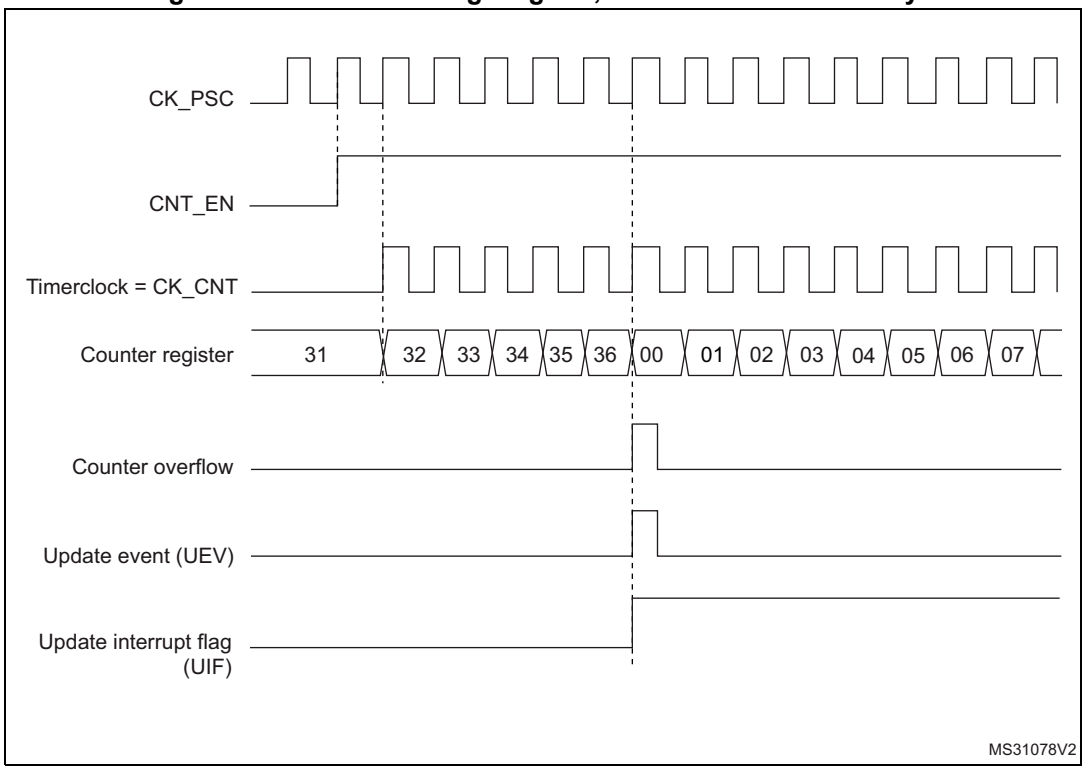


Figure 232. Counter timing diagram, internal clock divided by 2

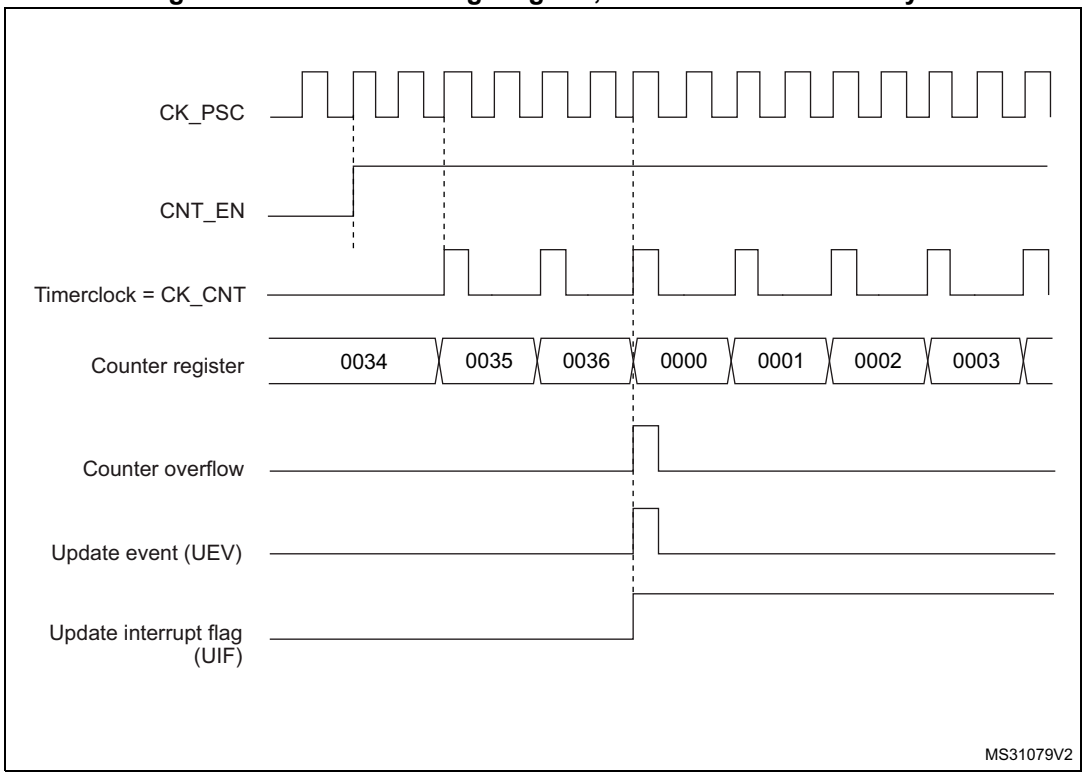


Figure 233. Counter timing diagram, internal clock divided by 4

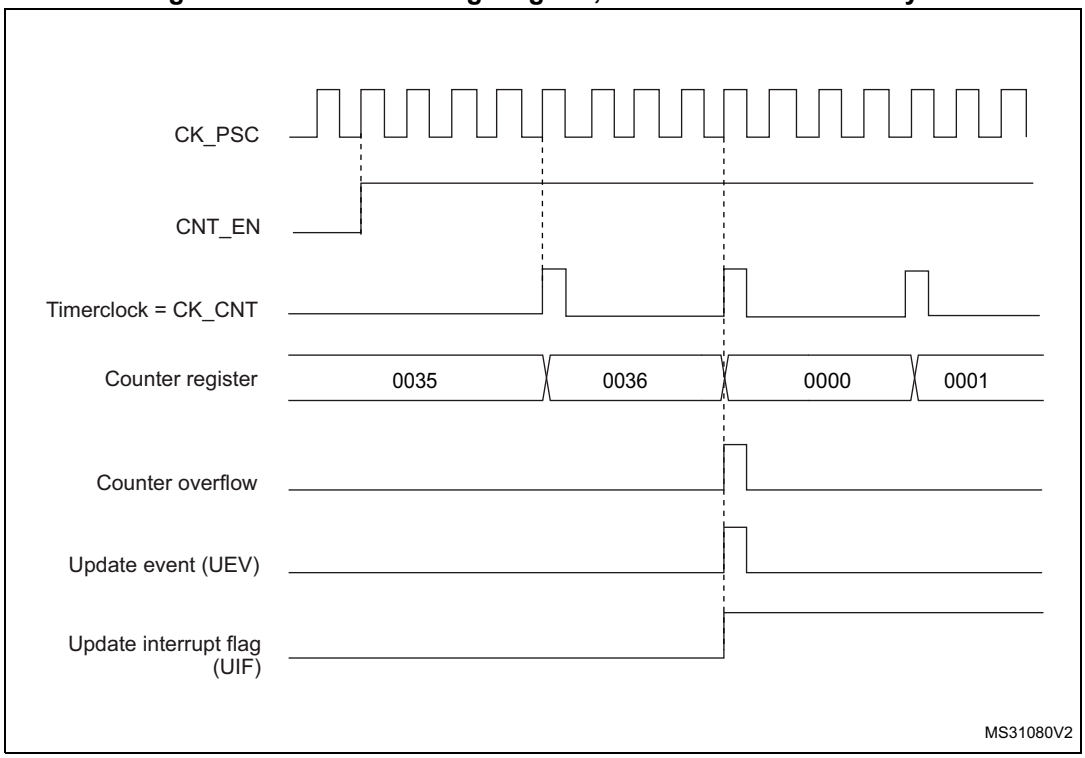


Figure 234. Counter timing diagram, internal clock divided by N

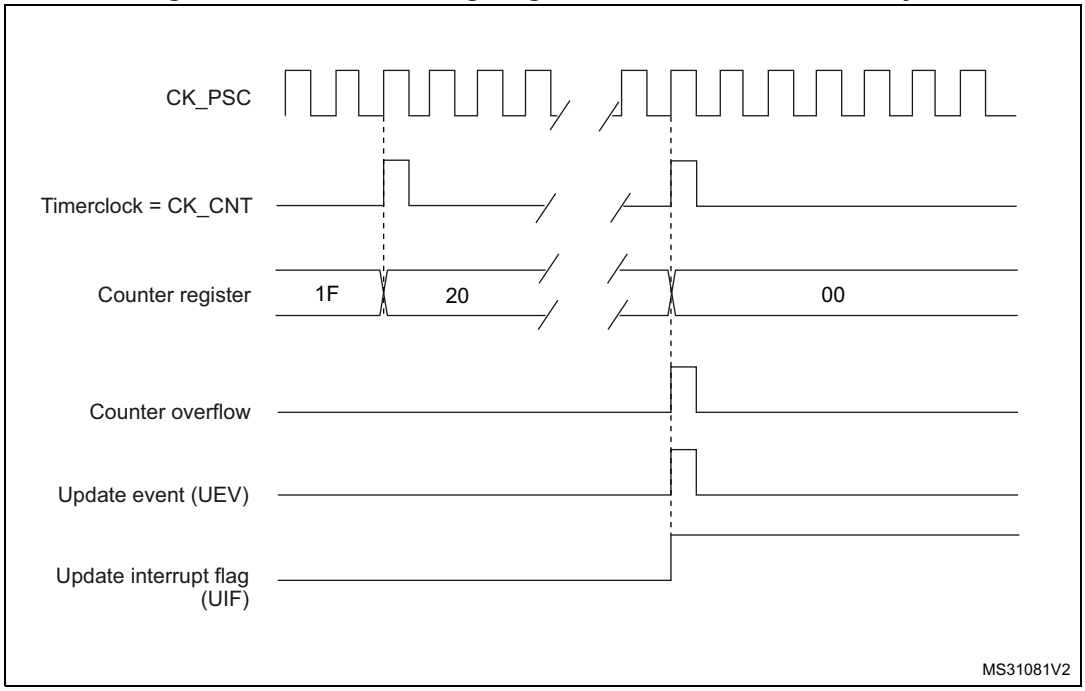


Figure 235. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

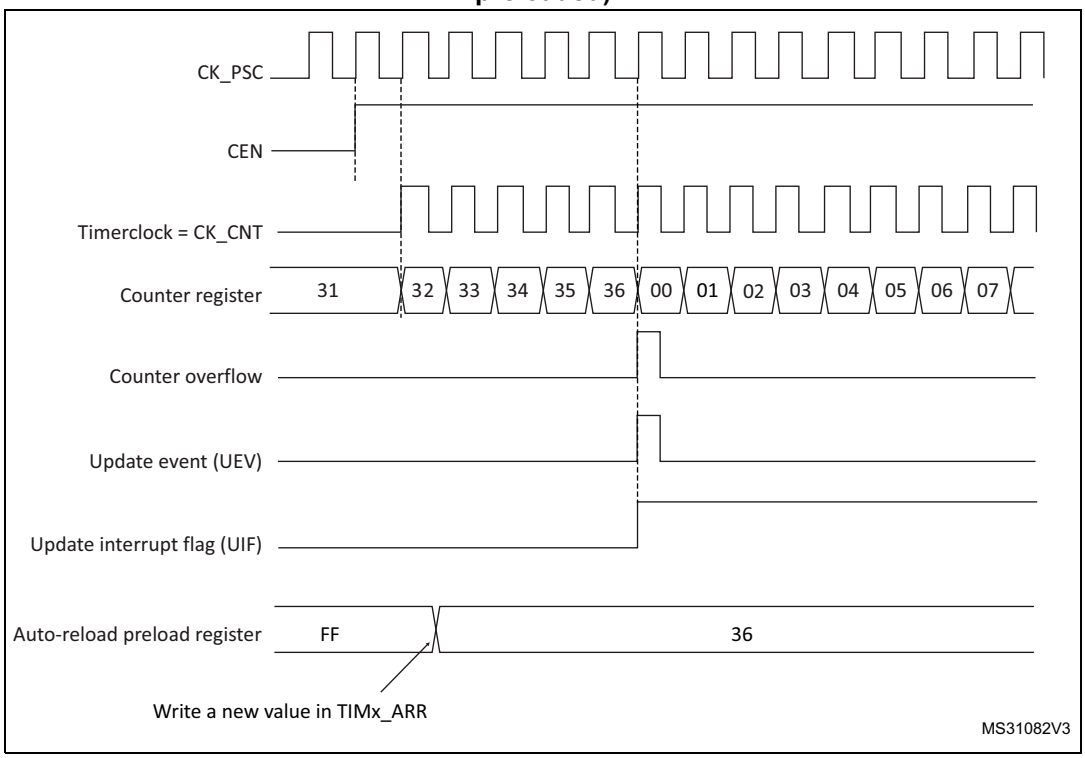
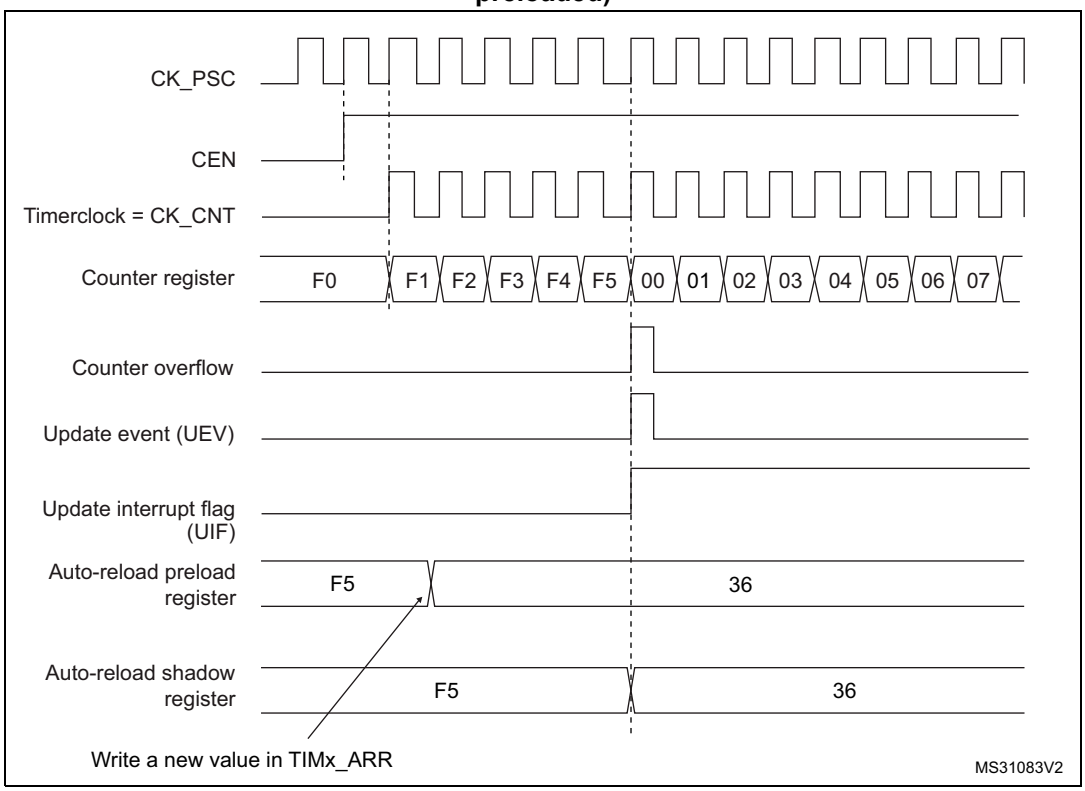


Figure 236. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



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 237. Counter timing diagram, internal clock divided by 1

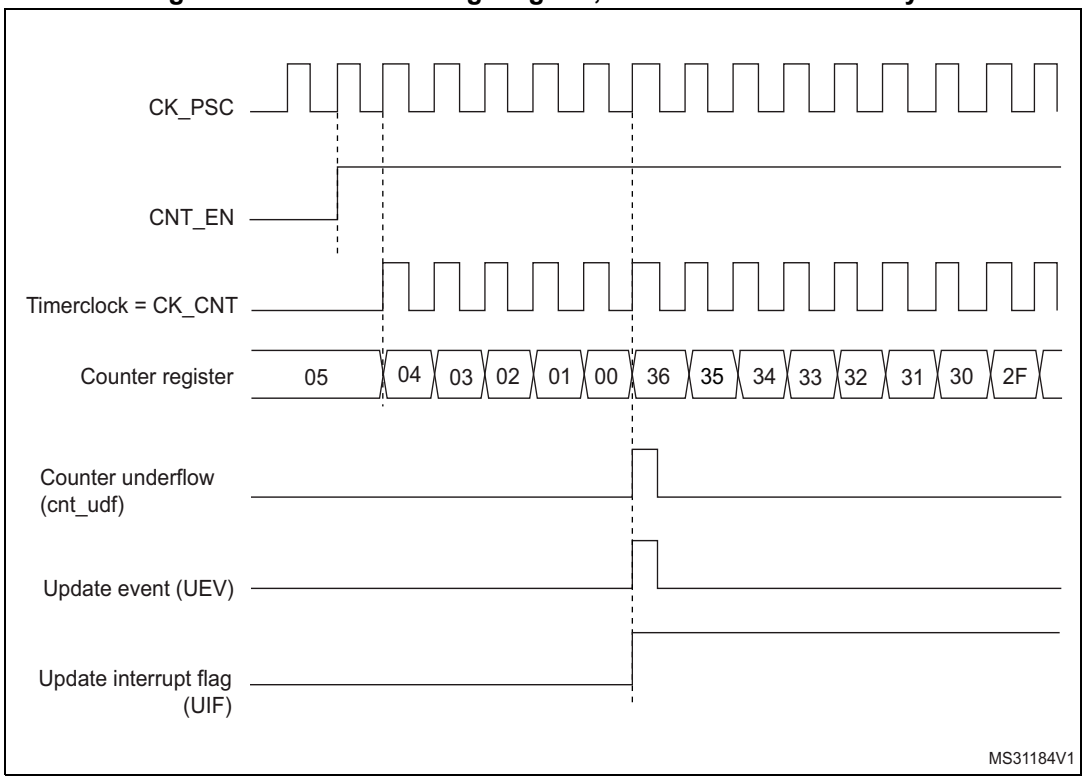


Figure 238. Counter timing diagram, internal clock divided by 2

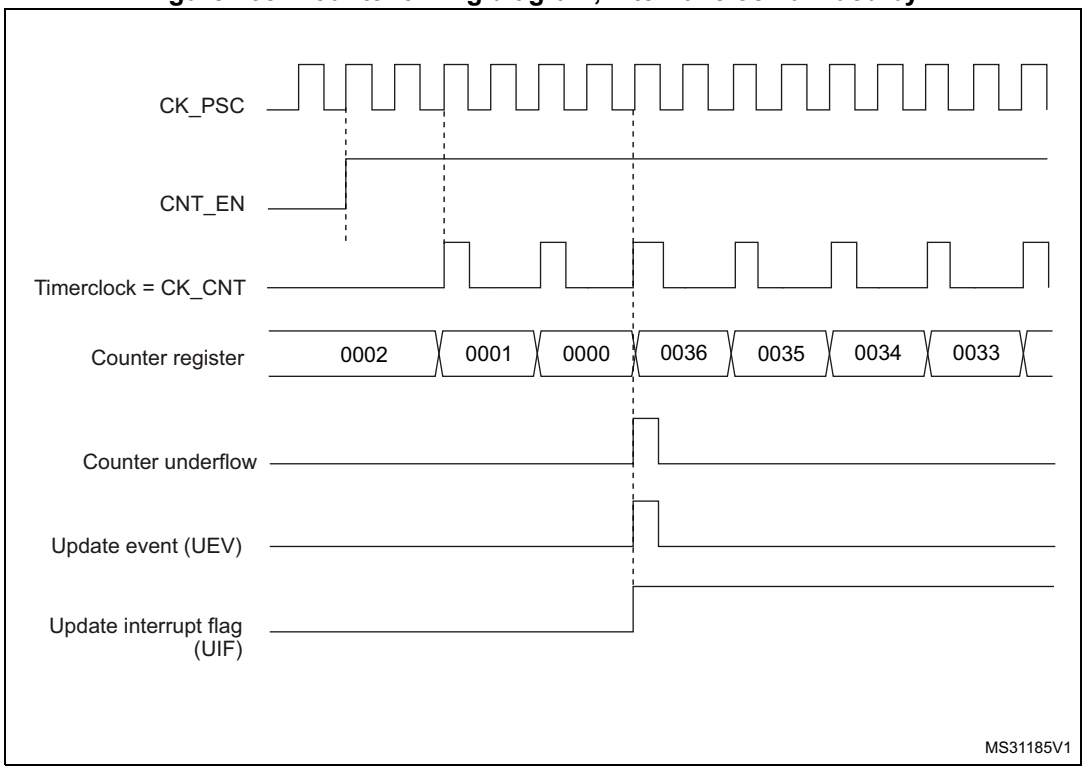


Figure 239. Counter timing diagram, internal clock divided by 4

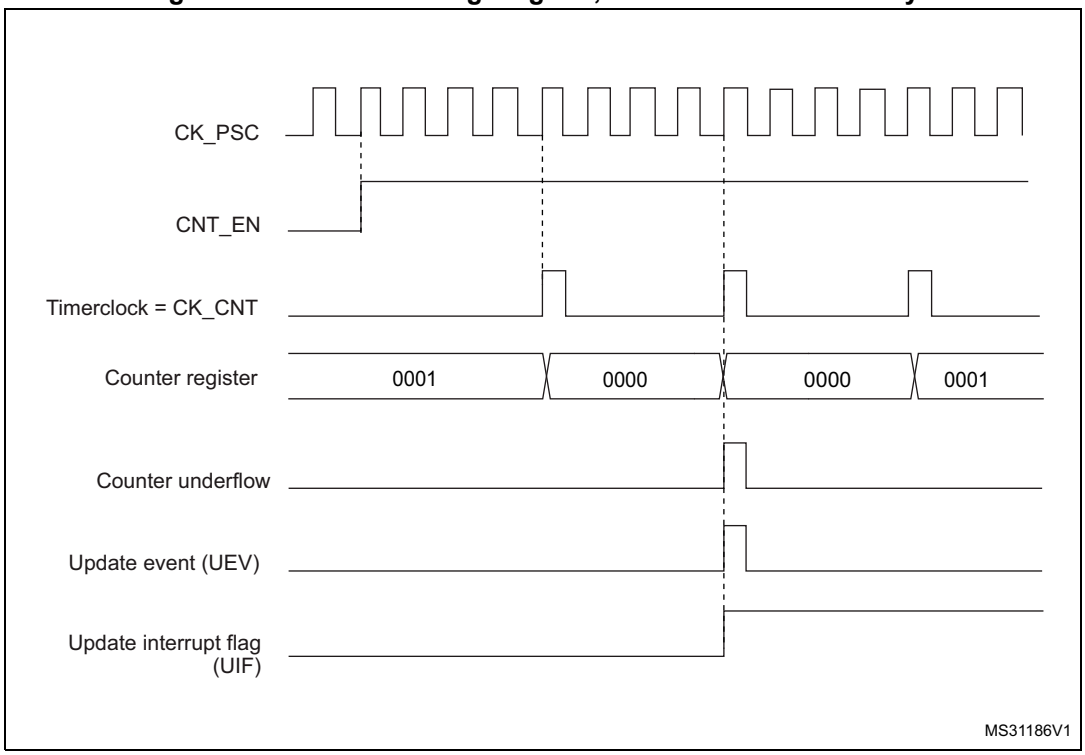


Figure 240. Counter timing diagram, internal clock divided by N

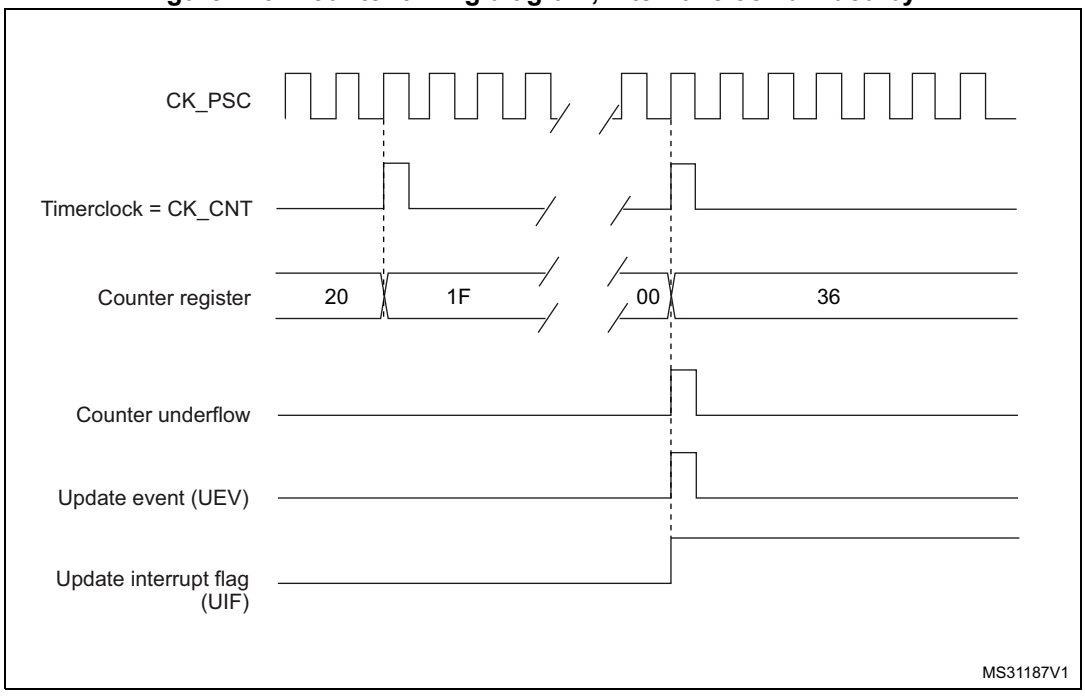
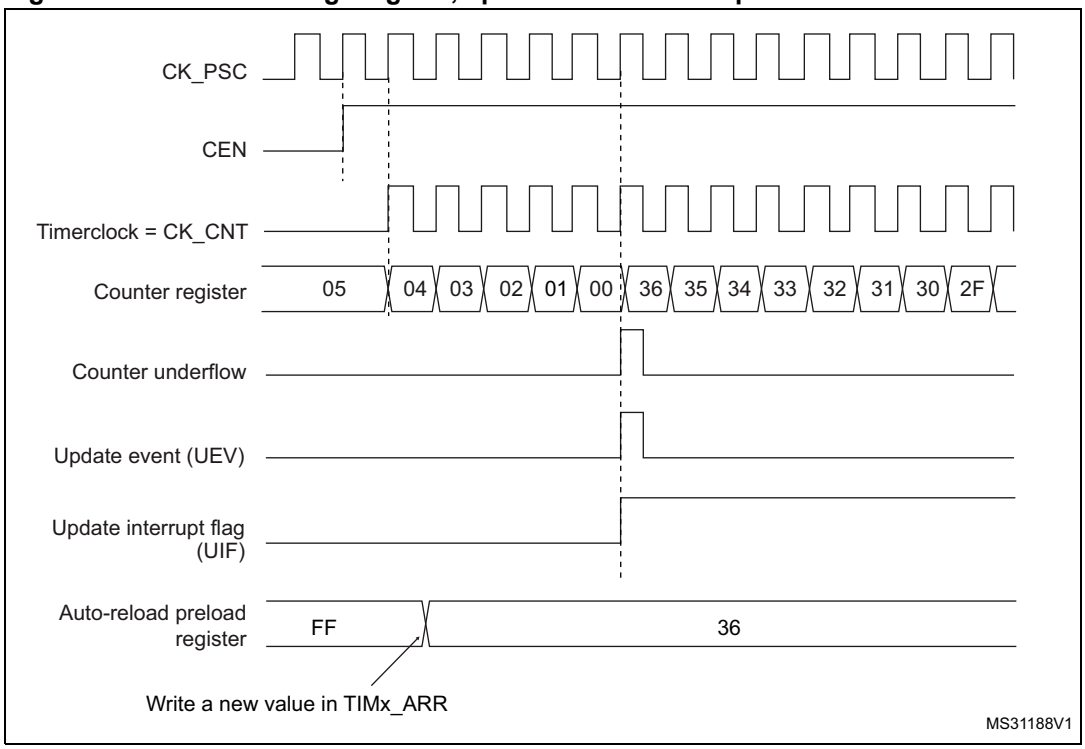


Figure 241. Counter timing diagram, update event when repetition counter is not used



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

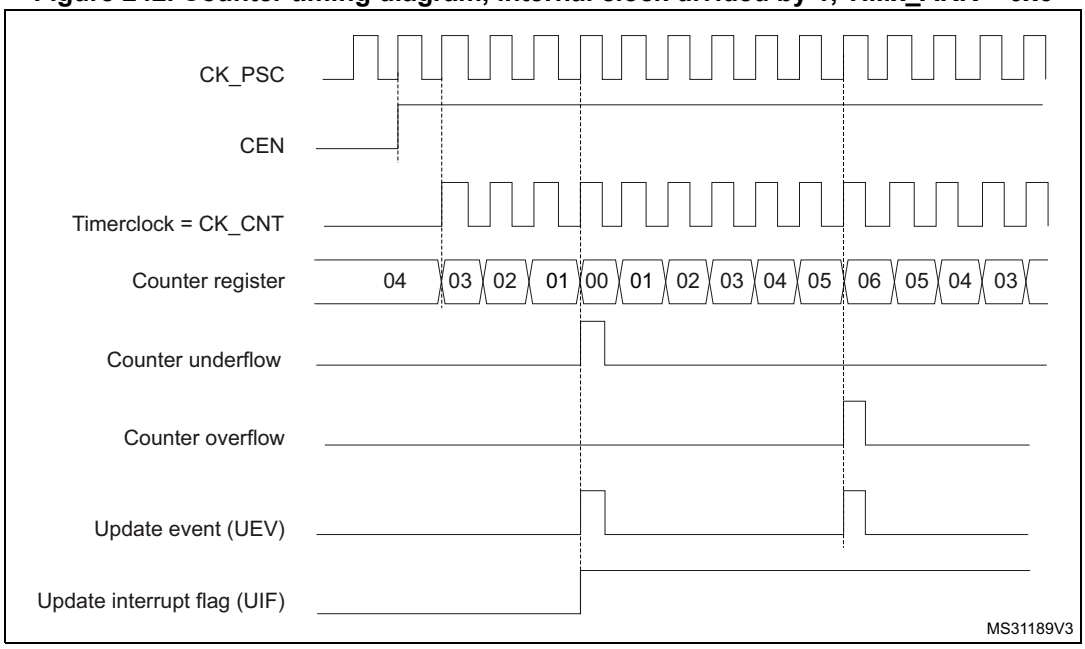
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

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 242. 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 33.4: TIM1/TIM8 registers*).

Figure 243. Counter timing diagram, internal clock divided by 2

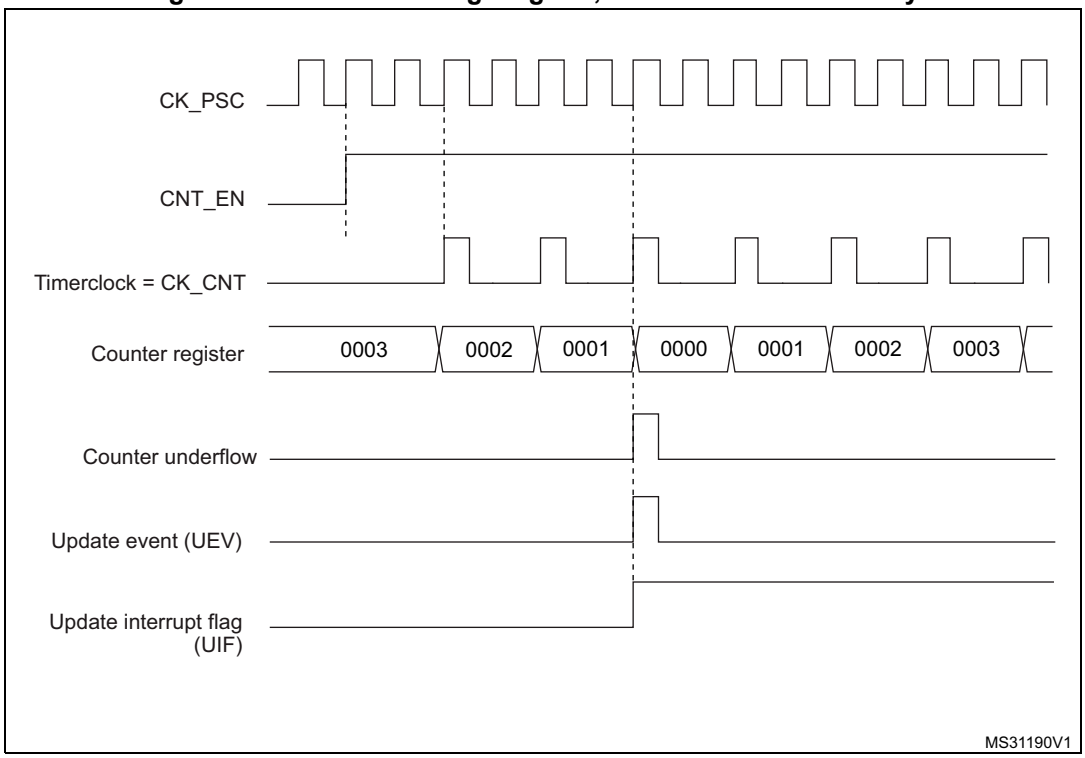


Figure 244. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

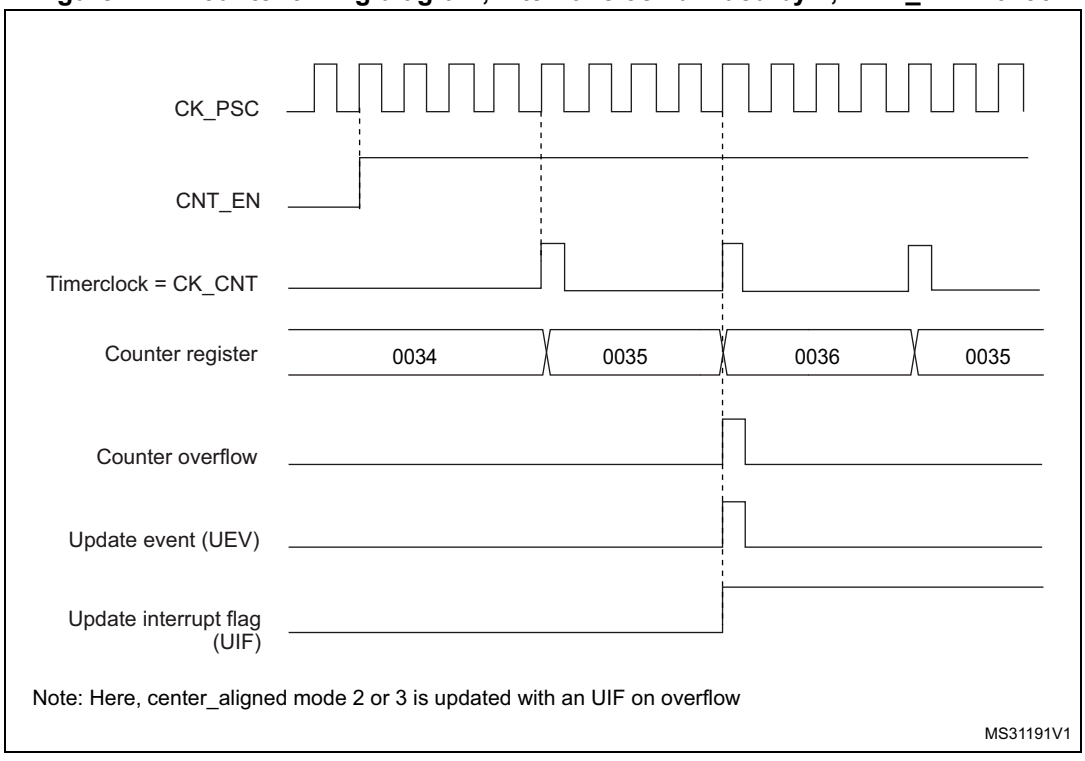


Figure 245. Counter timing diagram, internal clock divided by N

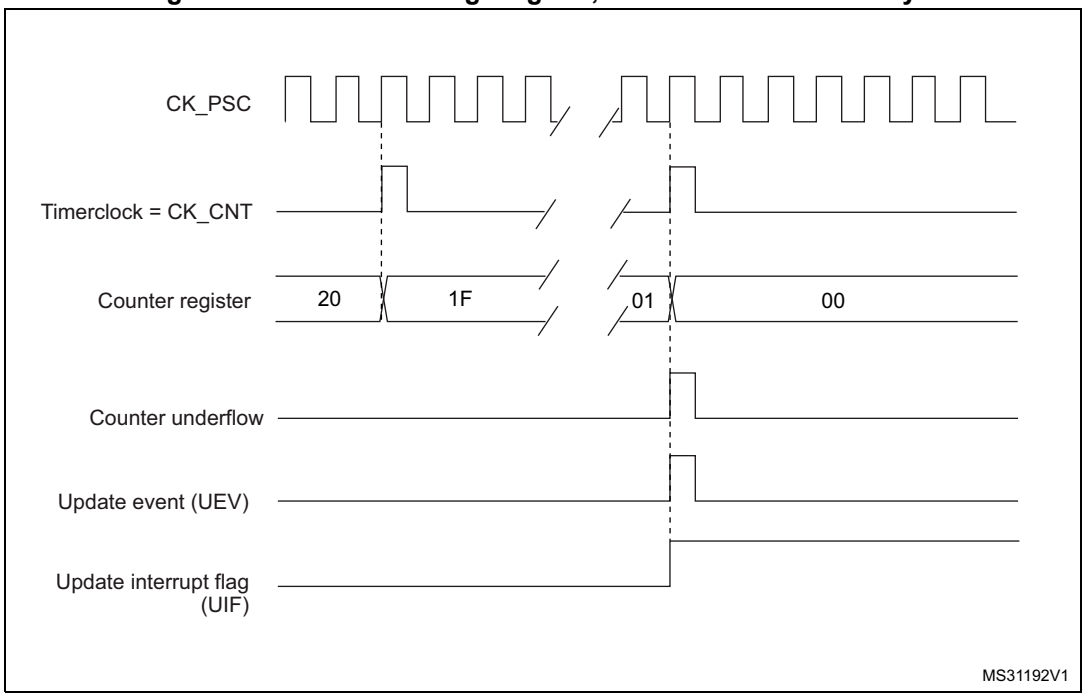


Figure 246. Counter timing diagram, update event with ARPE=1 (counter underflow)

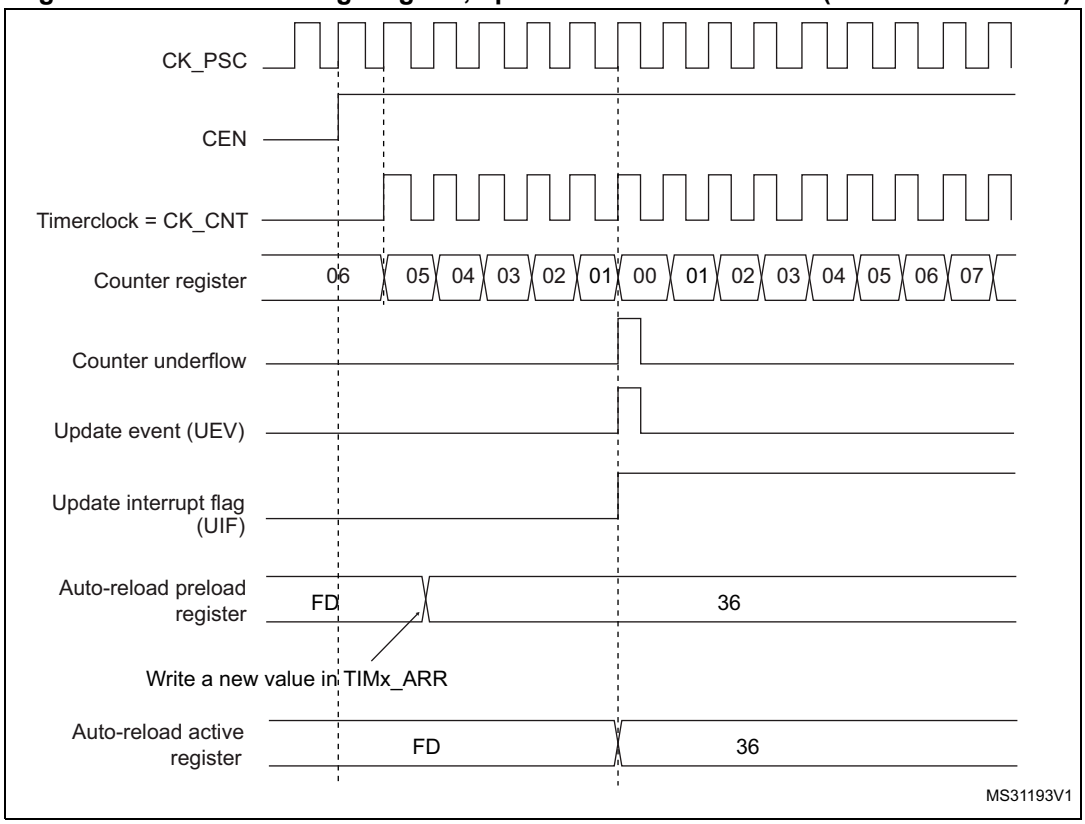
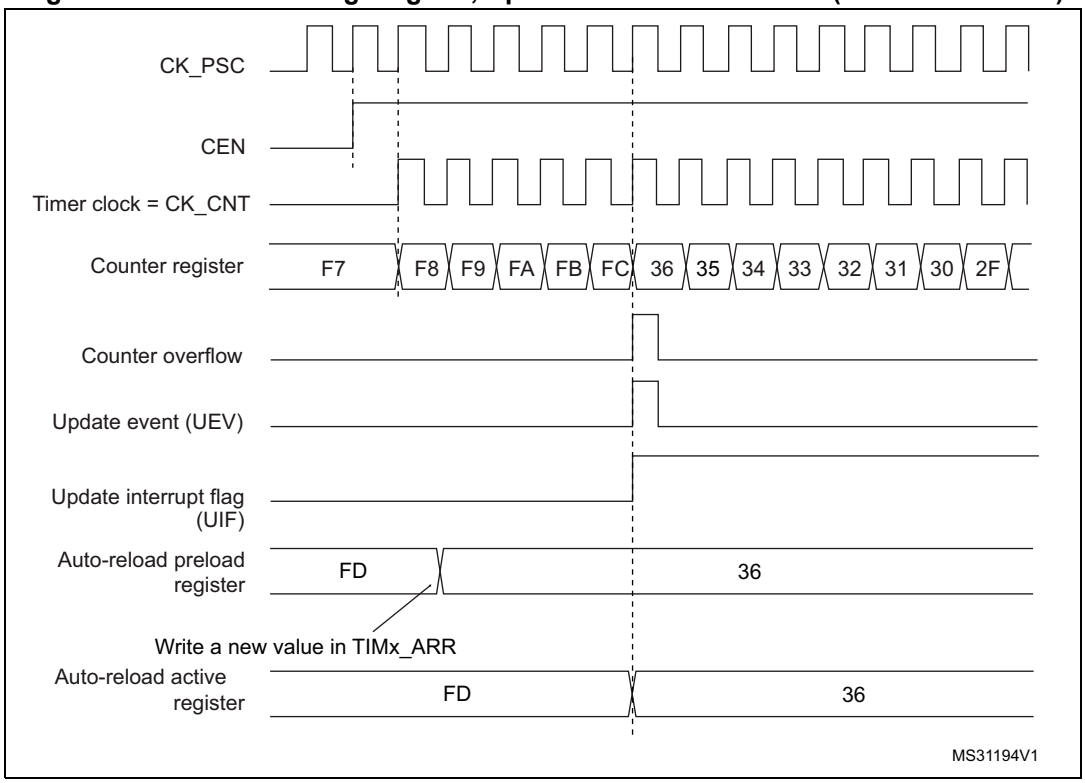


Figure 247. Counter timing diagram, Update event with ARPE=1 (counter overflow)



33.3.3 Repetition counter

Section 33.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

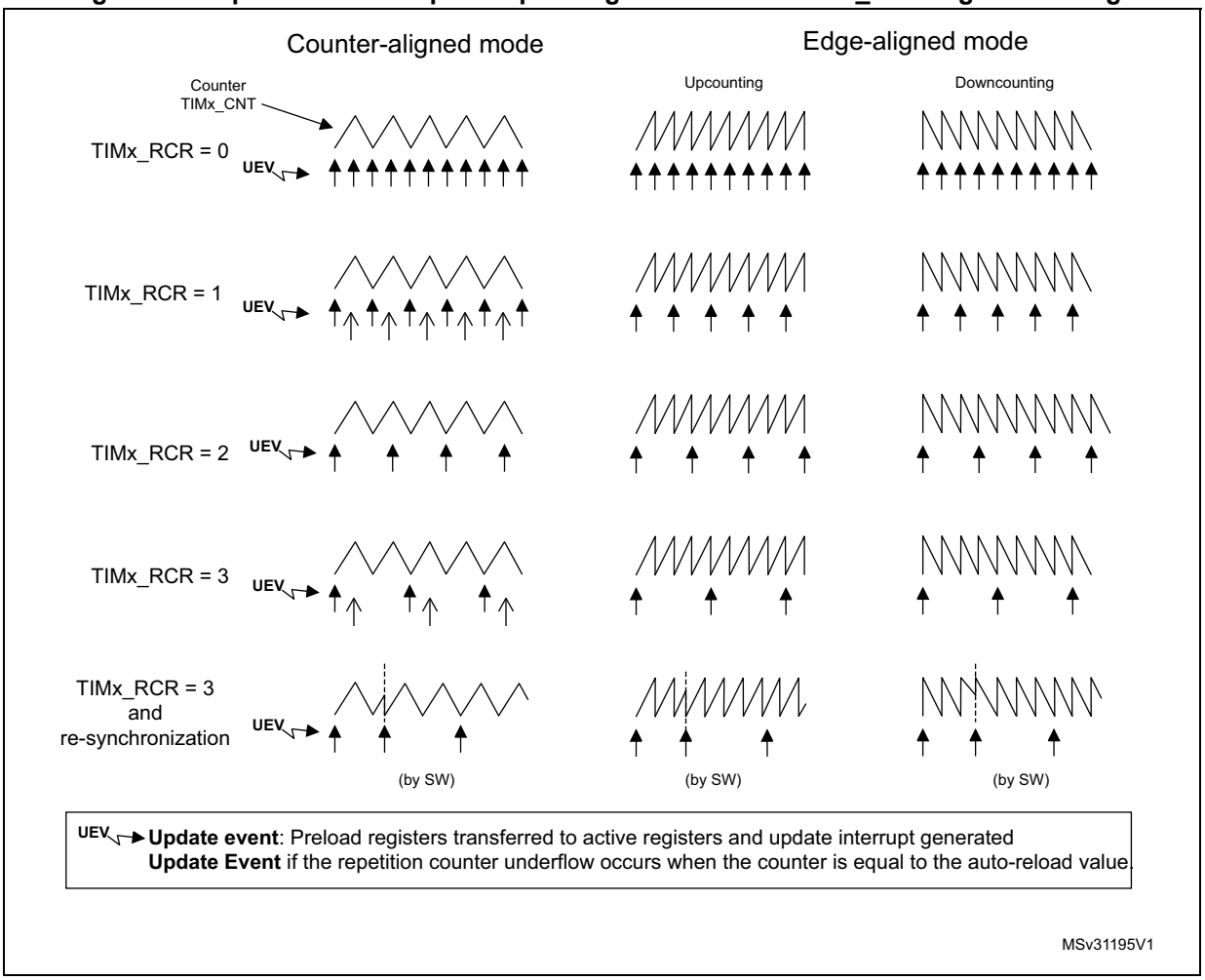
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 248*). 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 248. Update rate examples depending on mode and TIMx_RCR register settings



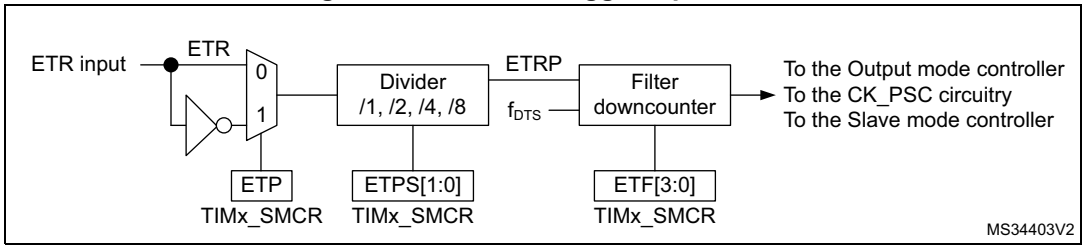
33.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see *Section 33.3.5*)
- trigger for the slave mode (see *Section 33.3.26*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 33.3.7*)

Figure 249 below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

Figure 249. External trigger input block



The ETR input comes from multiple sources: input pins (default configuration), comparator outputs and analog watchdogs. The selection is done with:

- the ETRSEL[2:0] bitfield in the TIMx_OR2 register
- the ETR_ADC1_RMP bitfield in the TIMxOR1[1:0] register

Figure 250. TIM1 ETR input circuitry

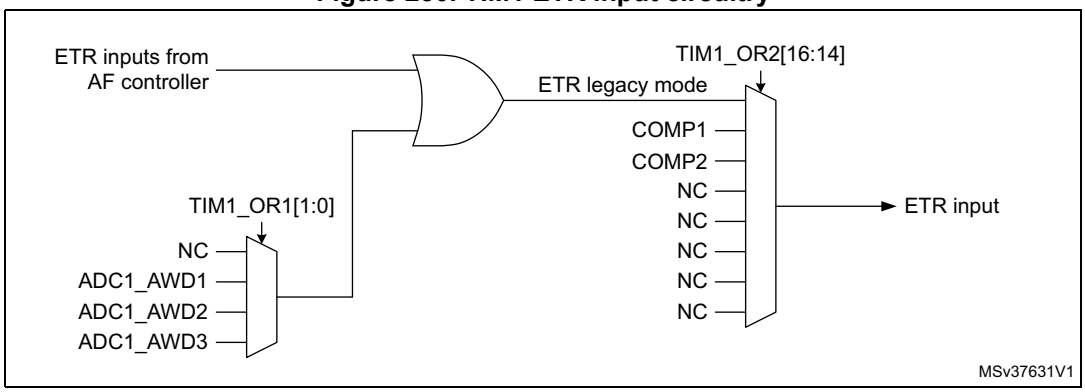
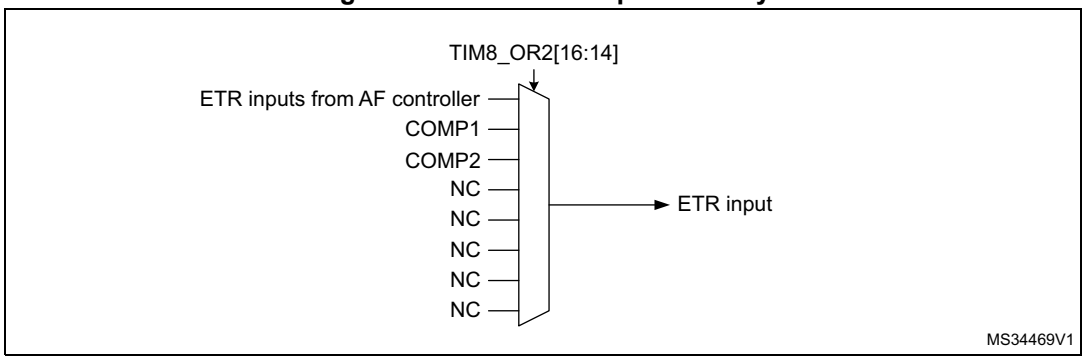


Figure 251. TIM8 ETR input circuitry



33.3.5 Clock selection

The counter clock can be provided by the following clock sources:

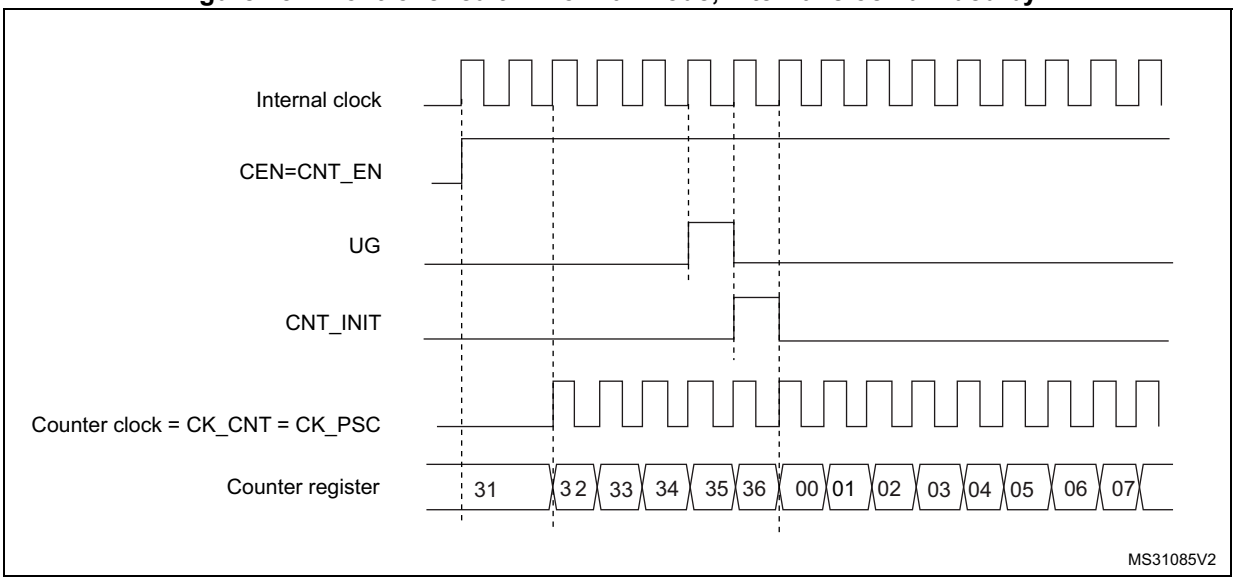
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 252 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

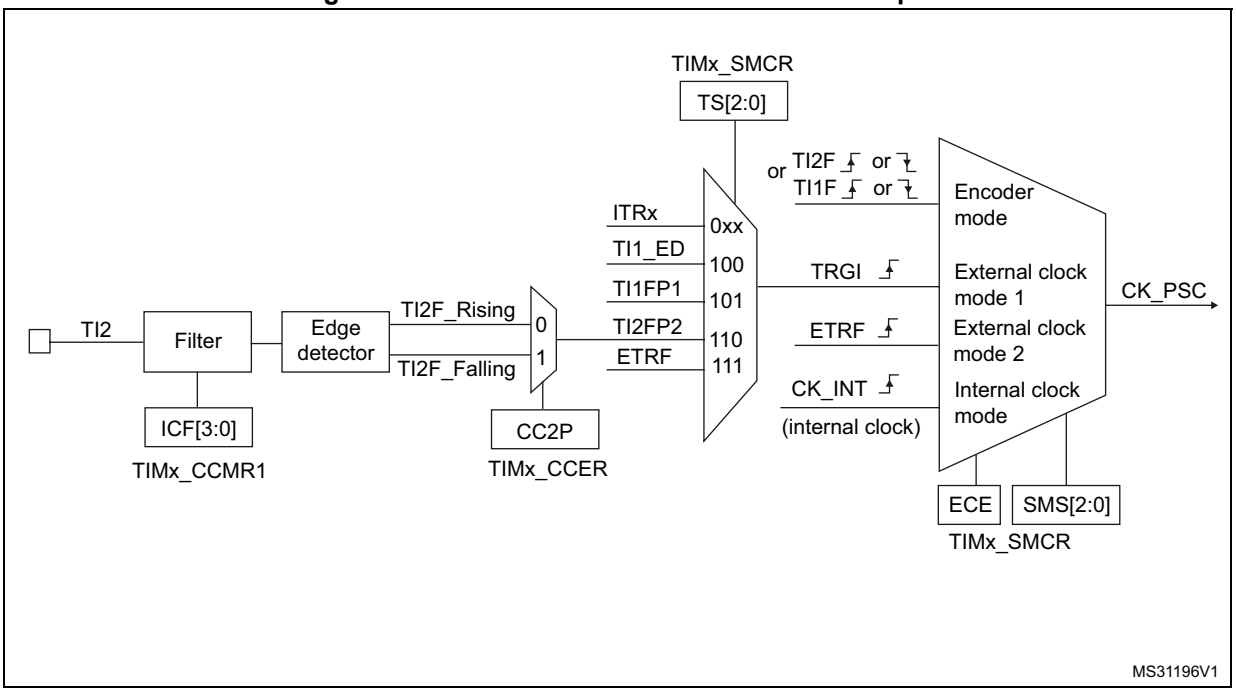
Figure 252. Control circuit in normal mode, internal clock divided by 1



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 253. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

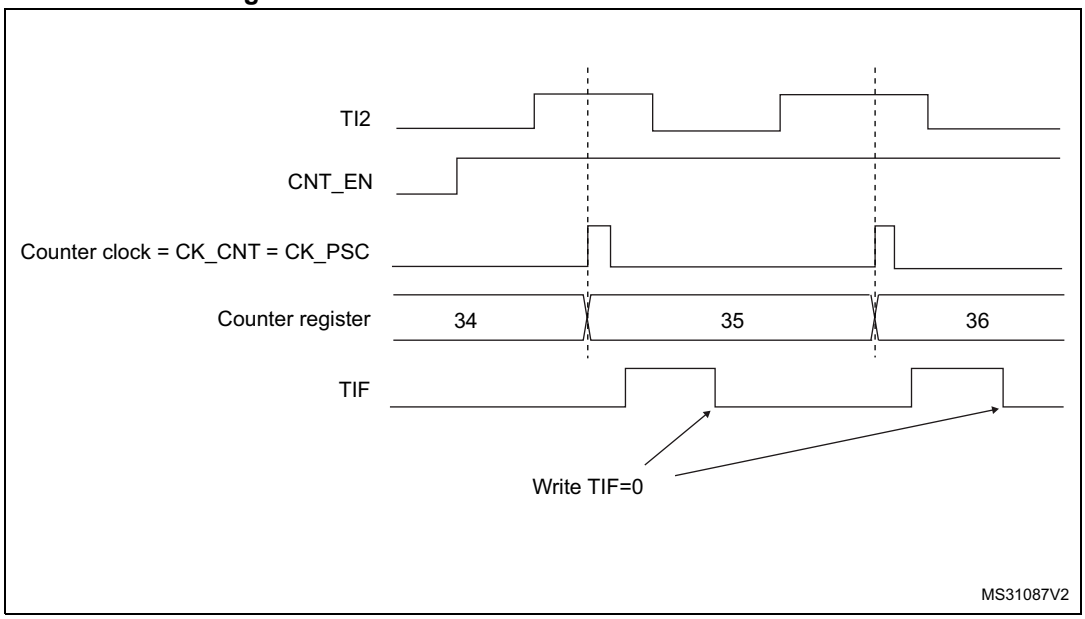
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 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 TI2 as the trigger input source by writing TS=110 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, so the user does not need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 254. Control circuit in external clock mode 1



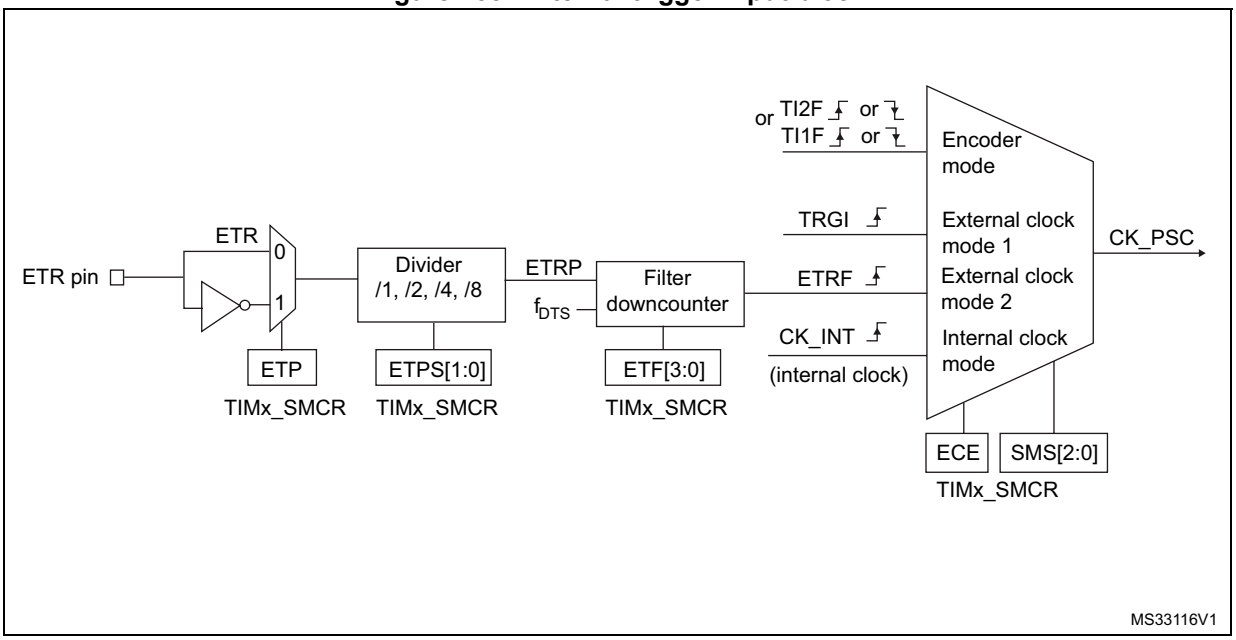
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The *Figure 255* gives an overview of the external trigger input block.

Figure 255. External trigger input block



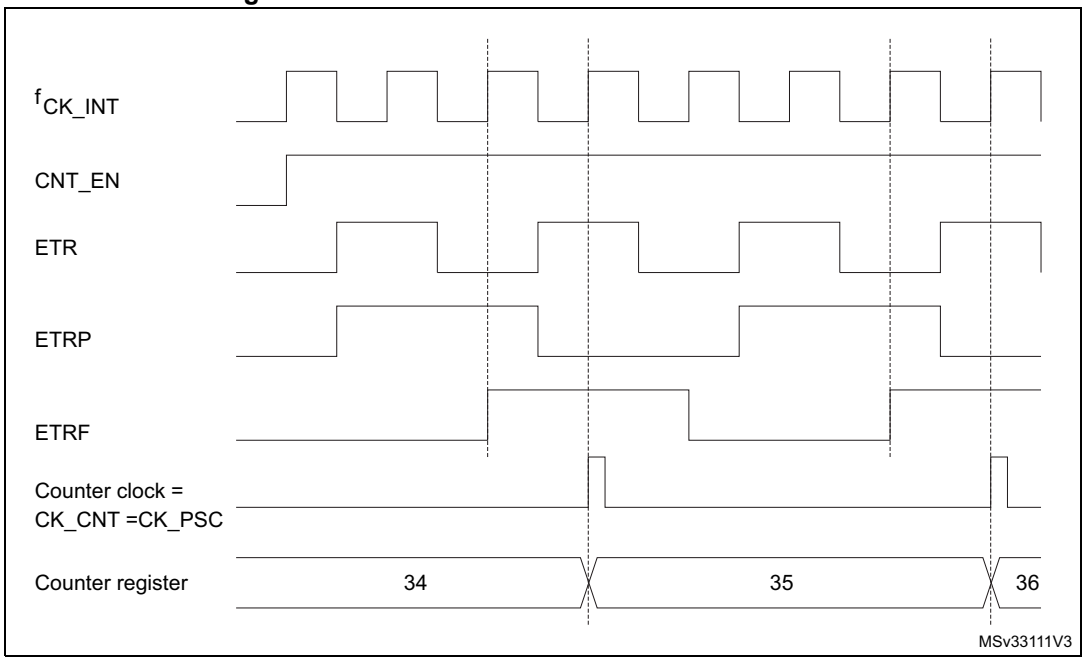
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by proper ETPS prescaler setting.

Figure 256. Control circuit in external clock mode 2



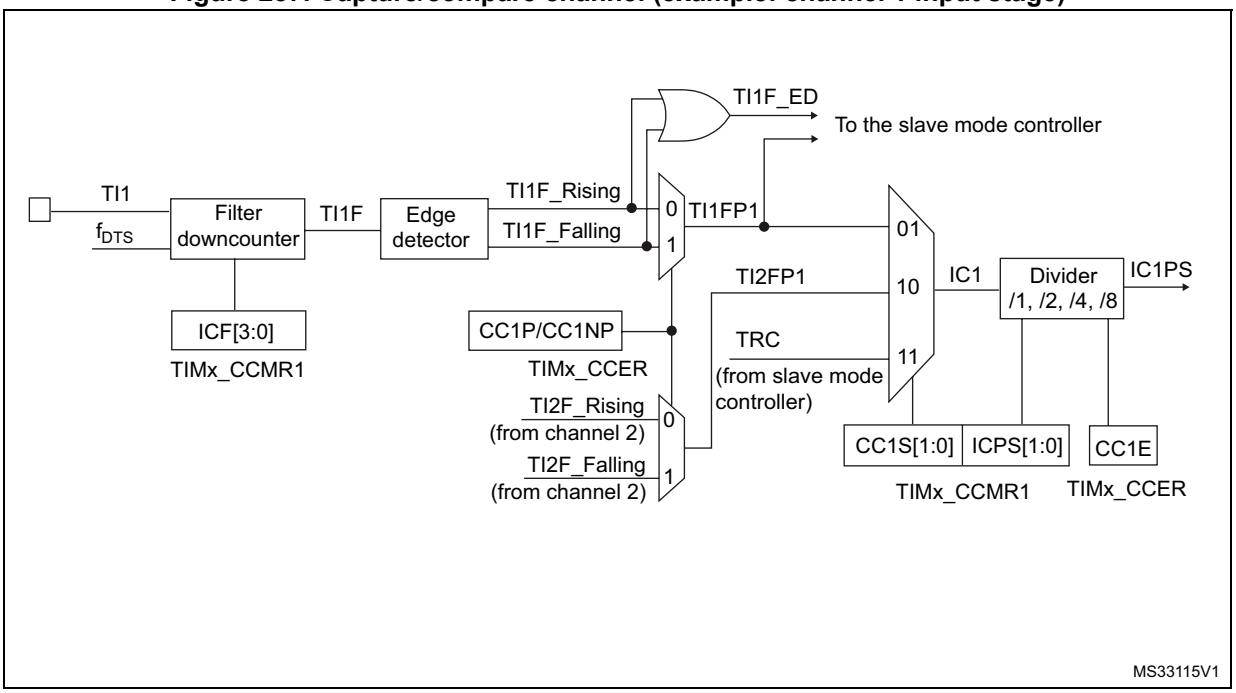
33.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 257 to *Figure 260* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 257. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 258. Capture/compare channel 1 main circuit

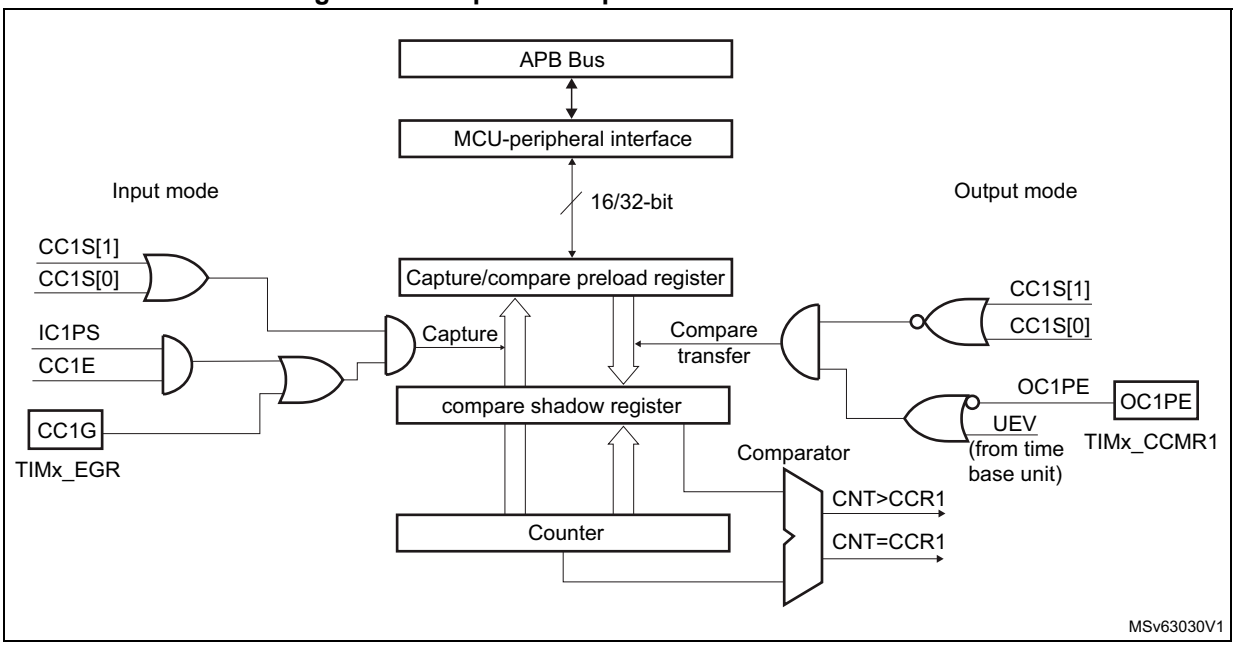
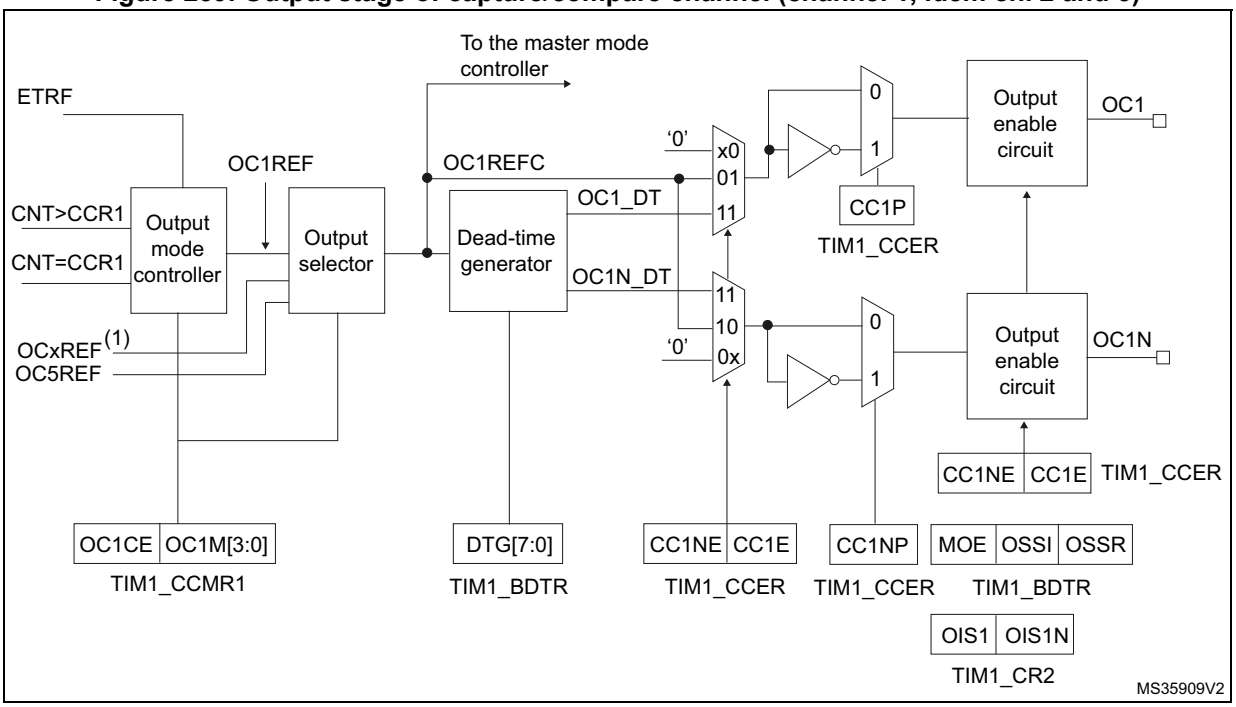


Figure 259. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)



1. OCxREF, where x is the rank of the complementary channel

Figure 260. Output stage of capture/compare channel (channel 4)

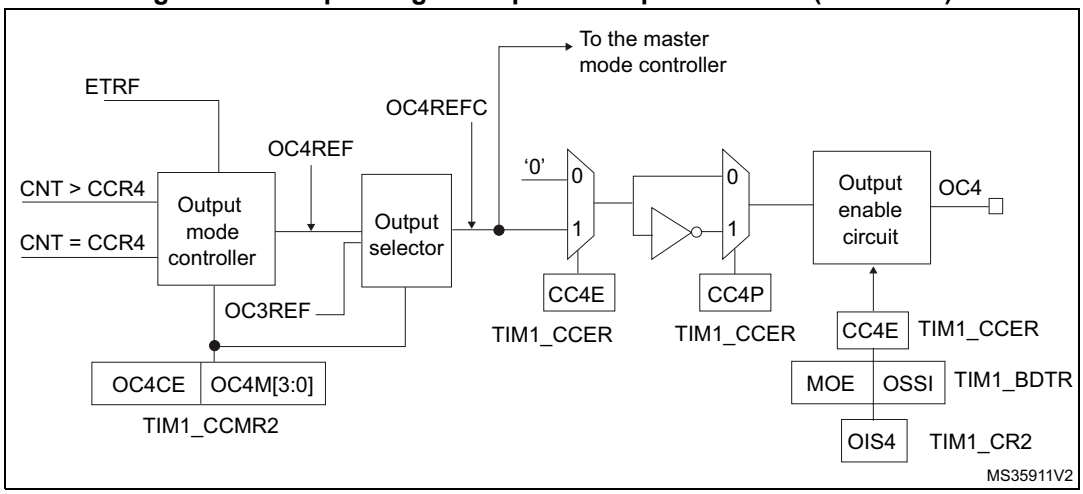
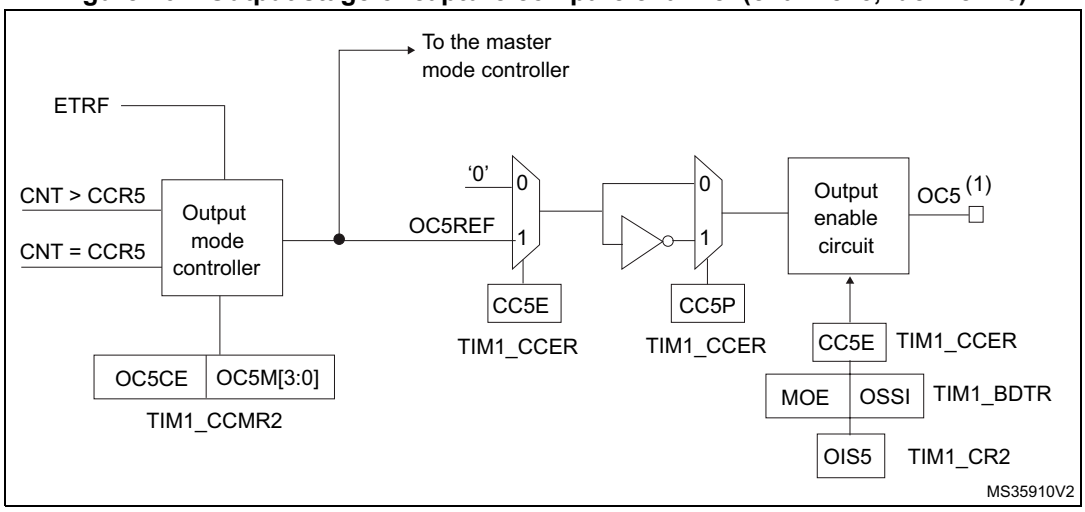


Figure 261. Output stage of capture/compare channel (channel 5, idem ch. 6)



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.

33.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be

cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written with ‘0’.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
3. Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
4. 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).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. 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.

33.3.8 PWM input mode

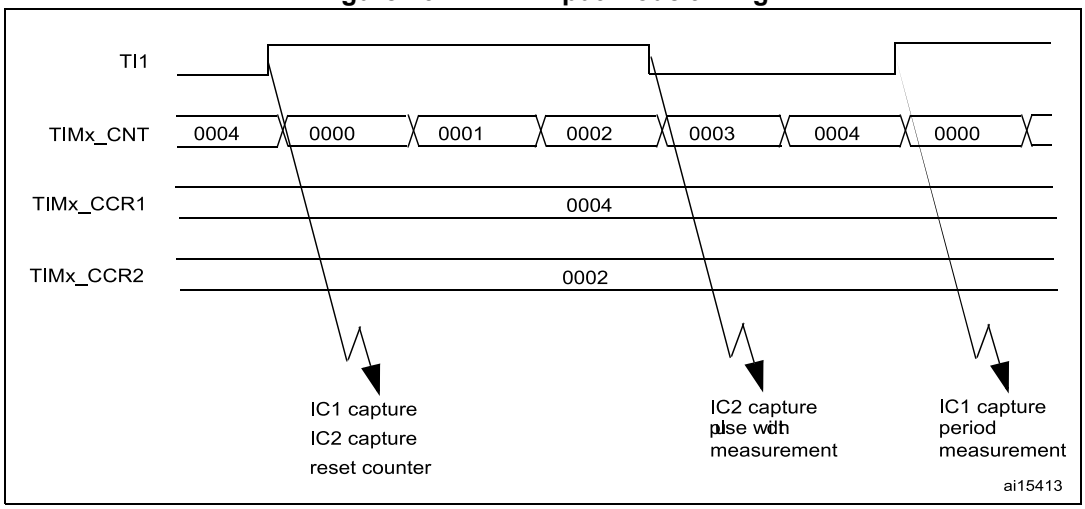
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, the user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP=’10’ (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 262. PWM input mode timing



33.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

33.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while Channel 5 and 6 are only available inside the device (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

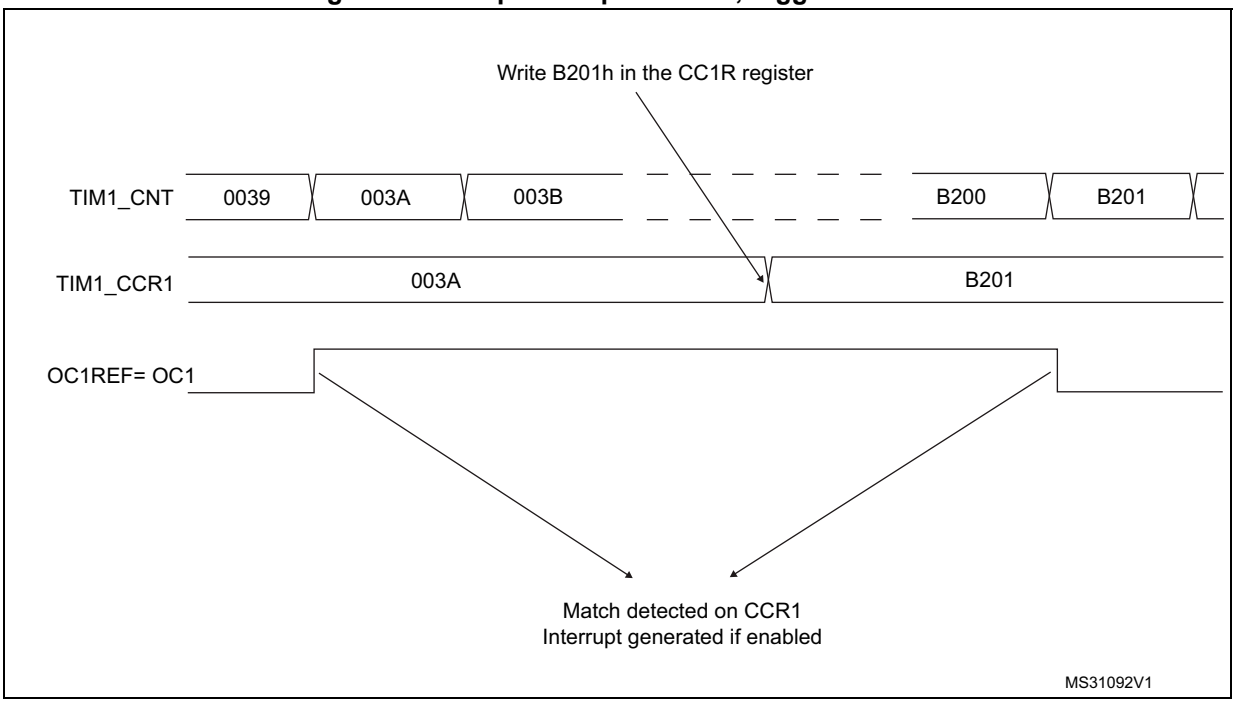
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 263*.

Figure 263. Output compare mode, toggle on OC1



33.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

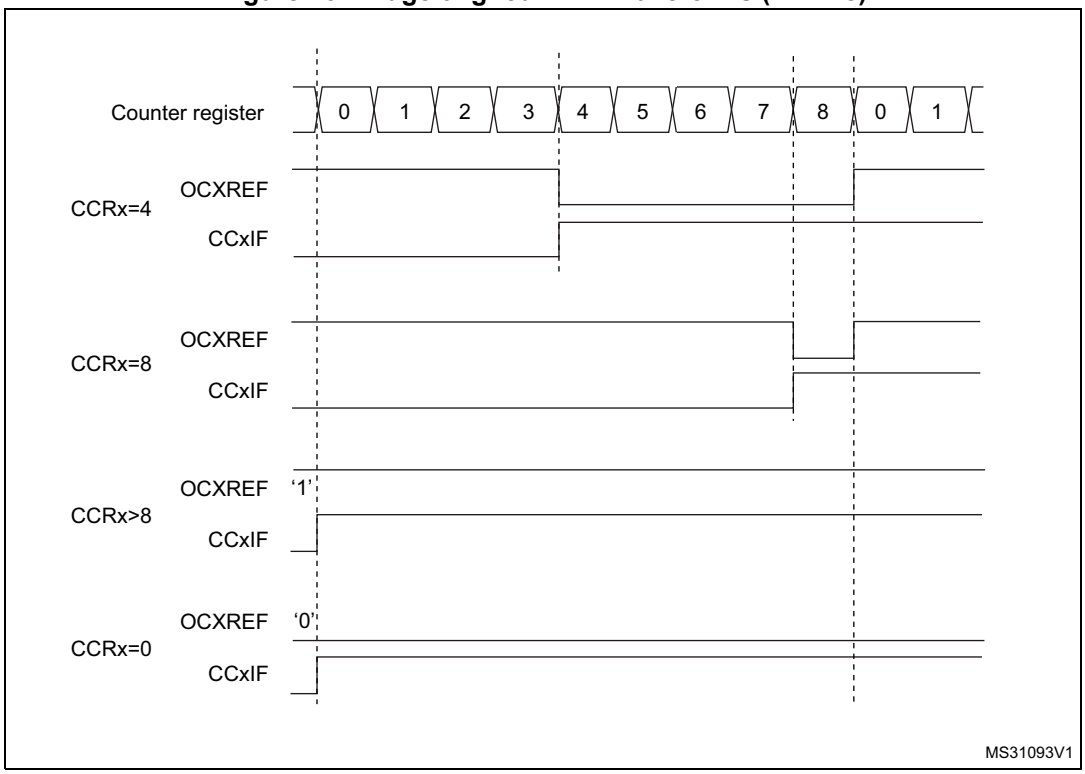
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 1080*.

  In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at '1'. If the compare value is 0 then OCxRef is held at '0'. *Figure 264* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 264. 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 1084*

  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at '1'. 0% PWM is not possible in this mode.

PWM center-aligned mode

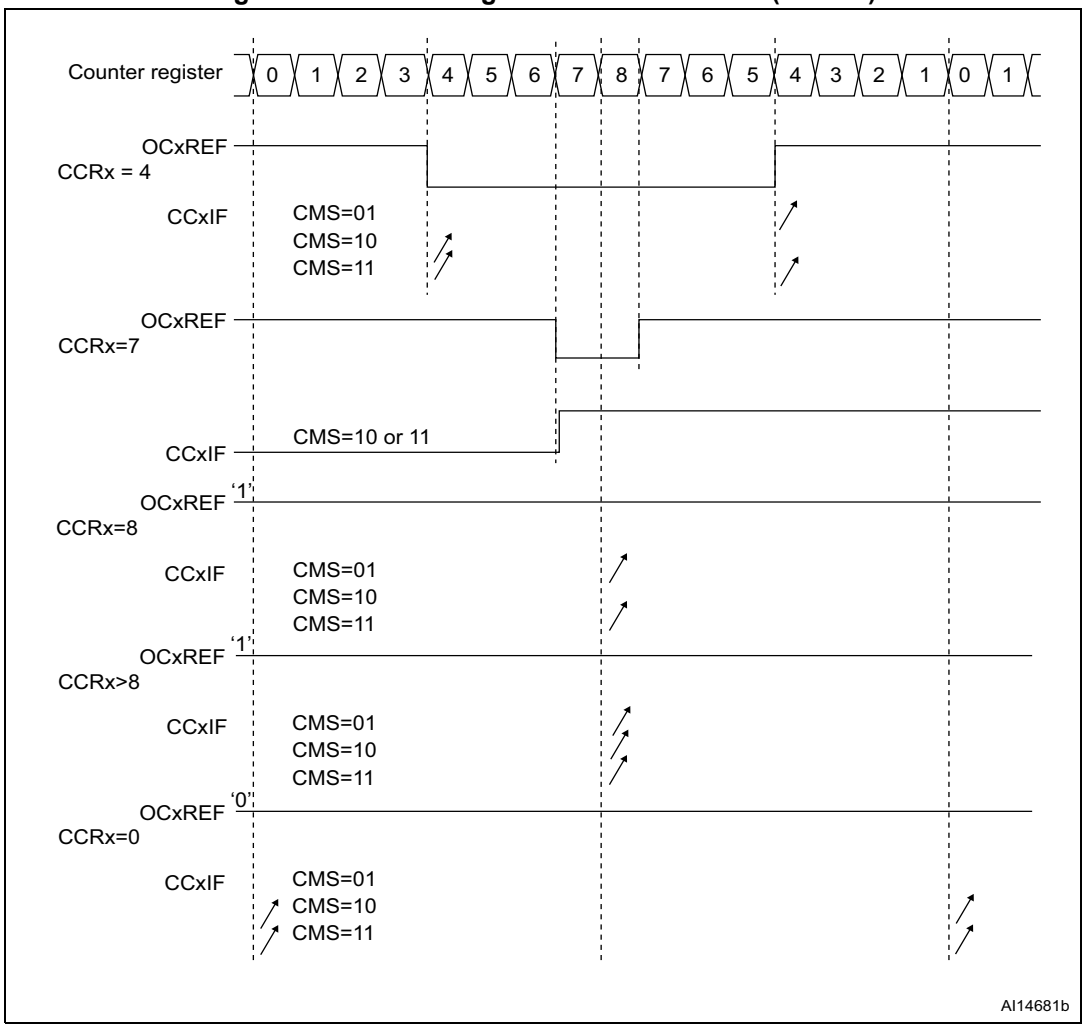
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the *Center-aligned mode (up/down counting) on page 1087*.

Figure 265 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 265. 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.

33.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

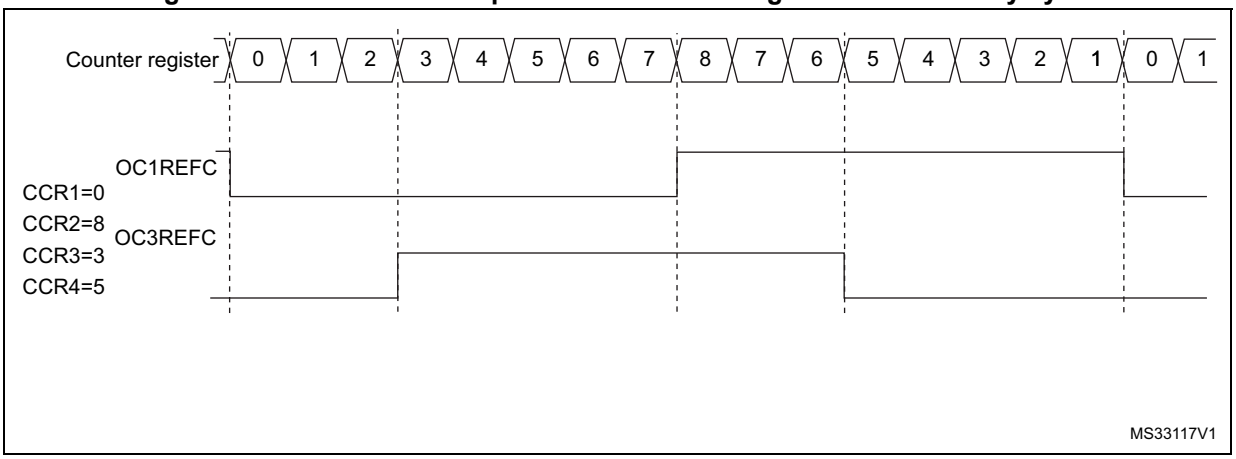
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

Figure 266 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 266. Generation of 2 phase-shifted PWM signals with 50% duty cycle



33.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

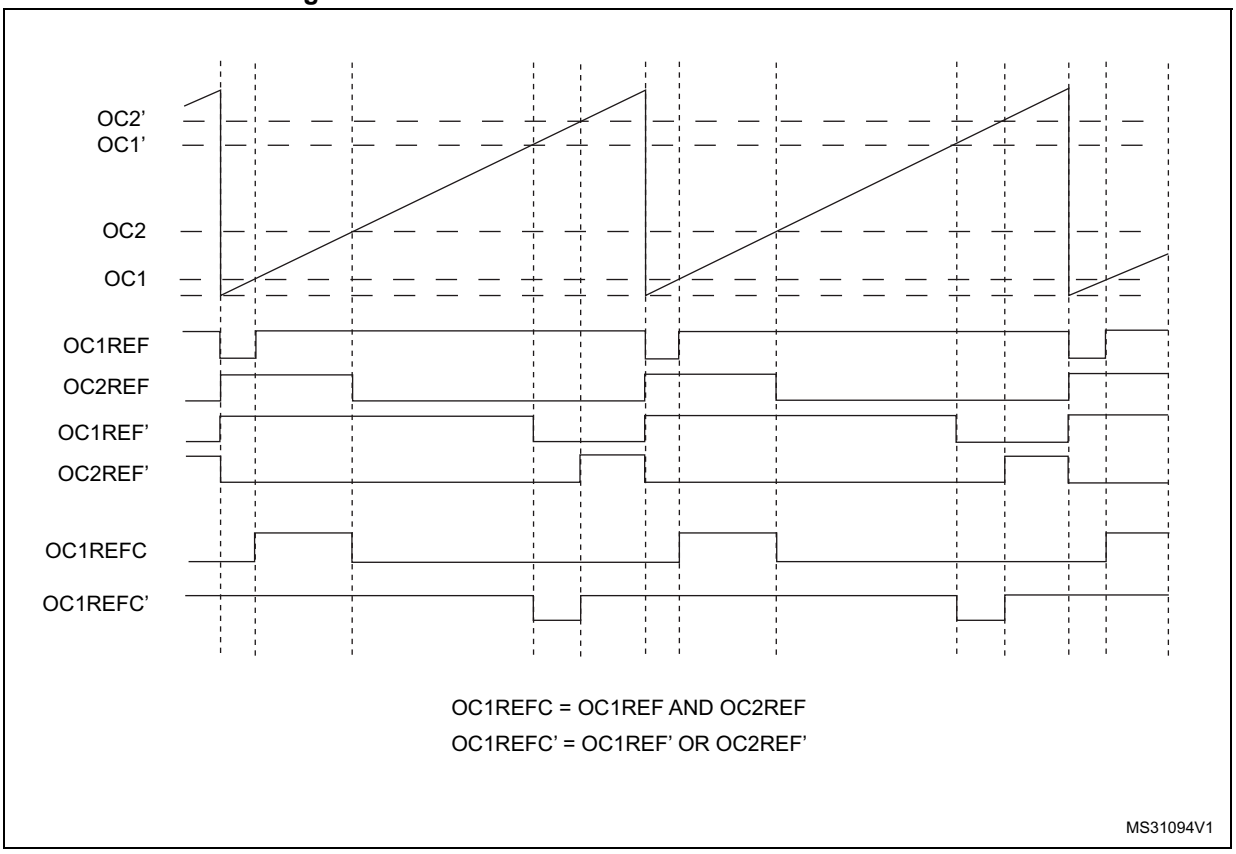
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 267 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.

Figure 267. Combined PWM mode on channel 1 and 3



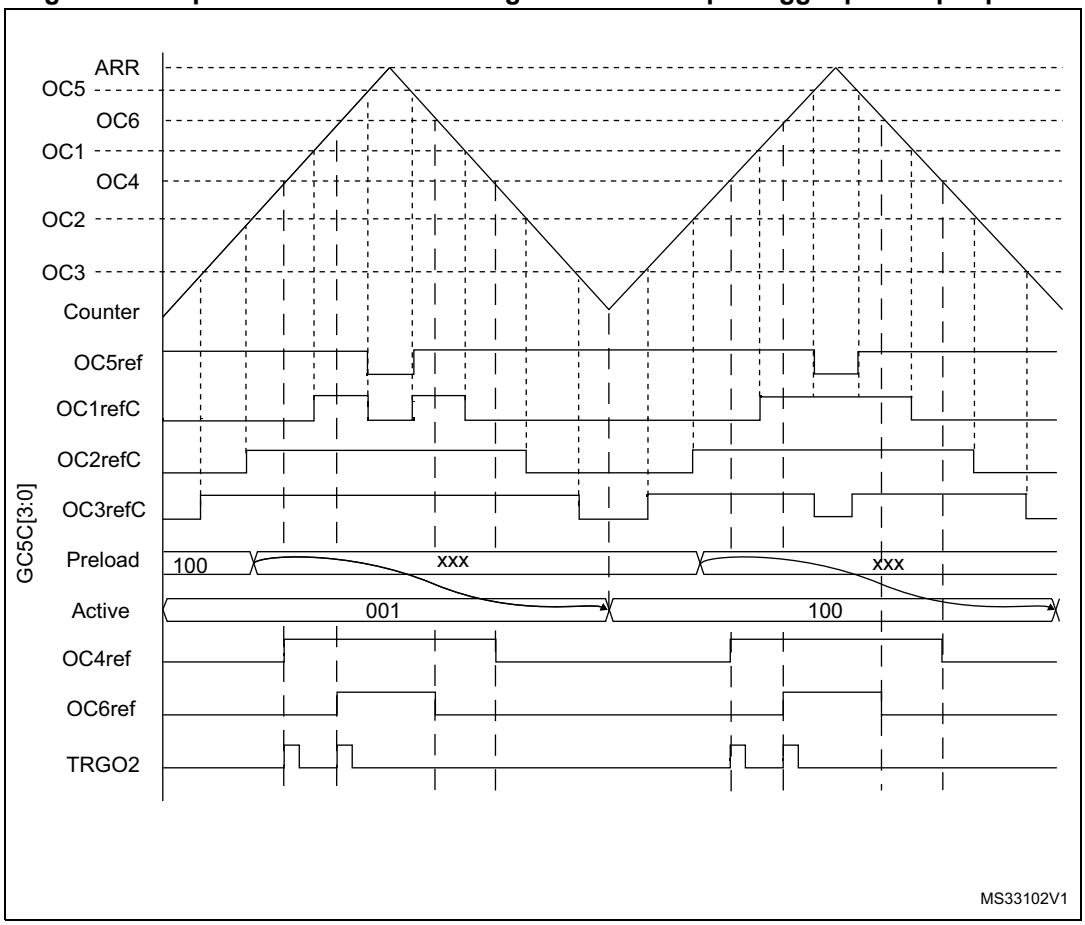
33.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, OC1REFC is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, OC2REFC is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, OC3REFC is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 268. 3-phase combined PWM signals with multiple trigger pulses per period



The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 33.3.27: ADC synchronization* for more details.

33.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 271: Output control bits for complementary OCx and OCxN channels with break feature on page 1156* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 269. Complementary output with dead-time insertion

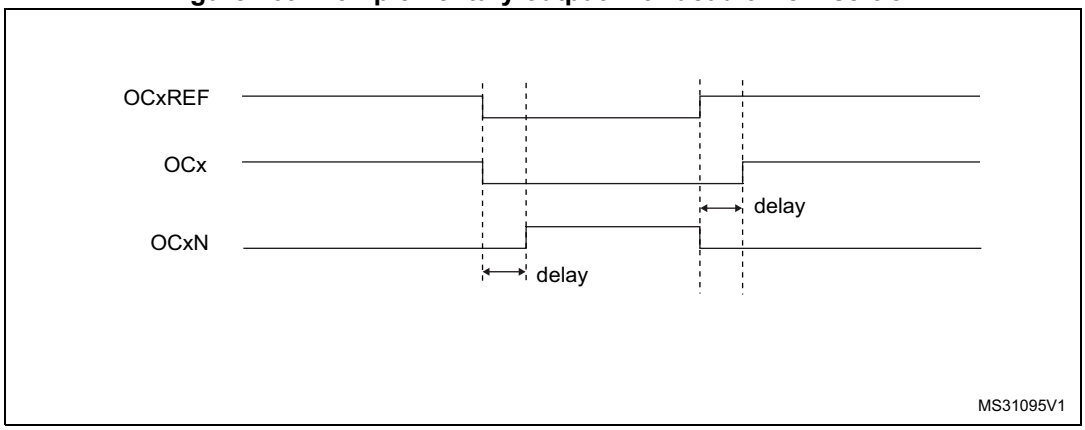


Figure 270. Dead-time waveforms with delay greater than the negative pulse

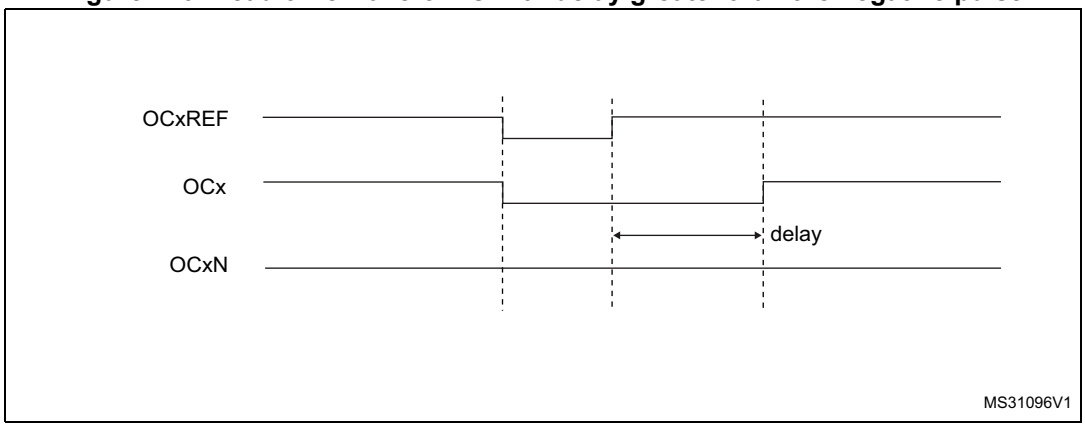
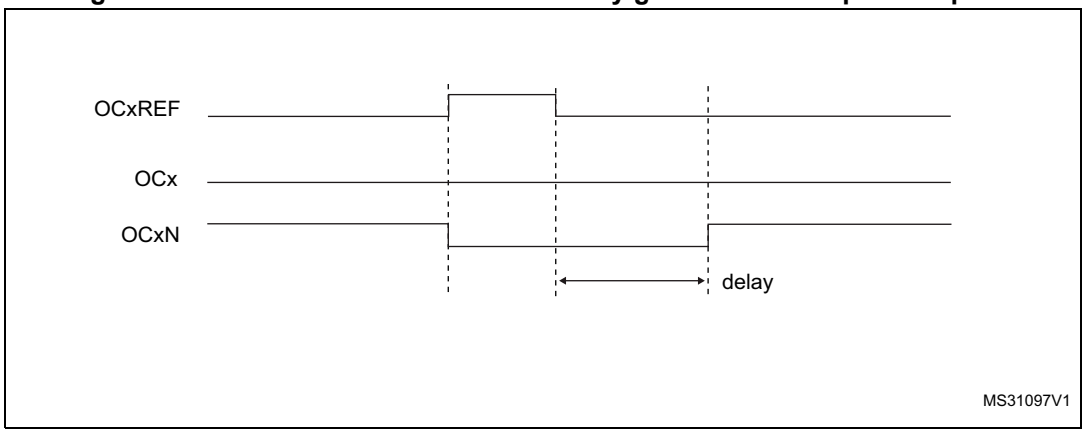


Figure 271. Dead-time waveforms with delay greater than the positive pulse



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 33.4.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

33.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 and TIM8 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, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows the outputs to be enabled/disabled by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 271: Output control bits for complementary OCx and OCxN channels with break feature on page 1156* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKE/BK2E and BKP/BK2P can be modified at the same time. When the BKE/BK2E and BKP/BK2P bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_OR2 and TIMx_OR3 registers.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the Cortex®-M33 LOCKUP output
  - the PVD output
  - the SRAM parity error signal
  - a Flash memory ECC dual error detection
  - a clock failure event generated by the CSS detector
  - the output from a comparator, with polarity selection and optional digital filtering
  - the analog watchdog output of the DFSDM1 peripheral

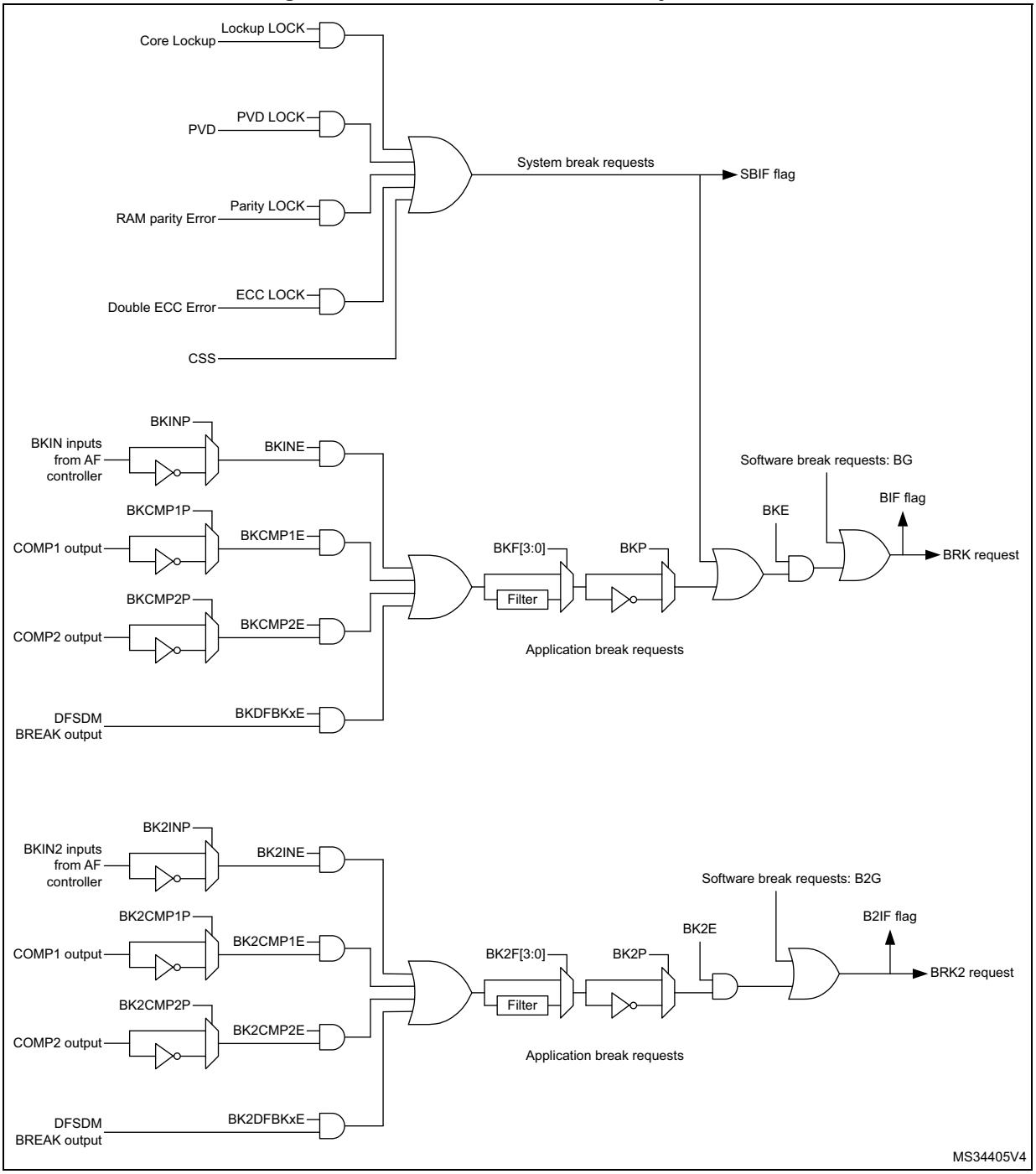
The sources for break2 (BRK2) are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source coming from a comparator output.

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.

All sources are ORed before entering the timer BRK or BRK2 inputs, as per *Figure 272* below.

Figure 272. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

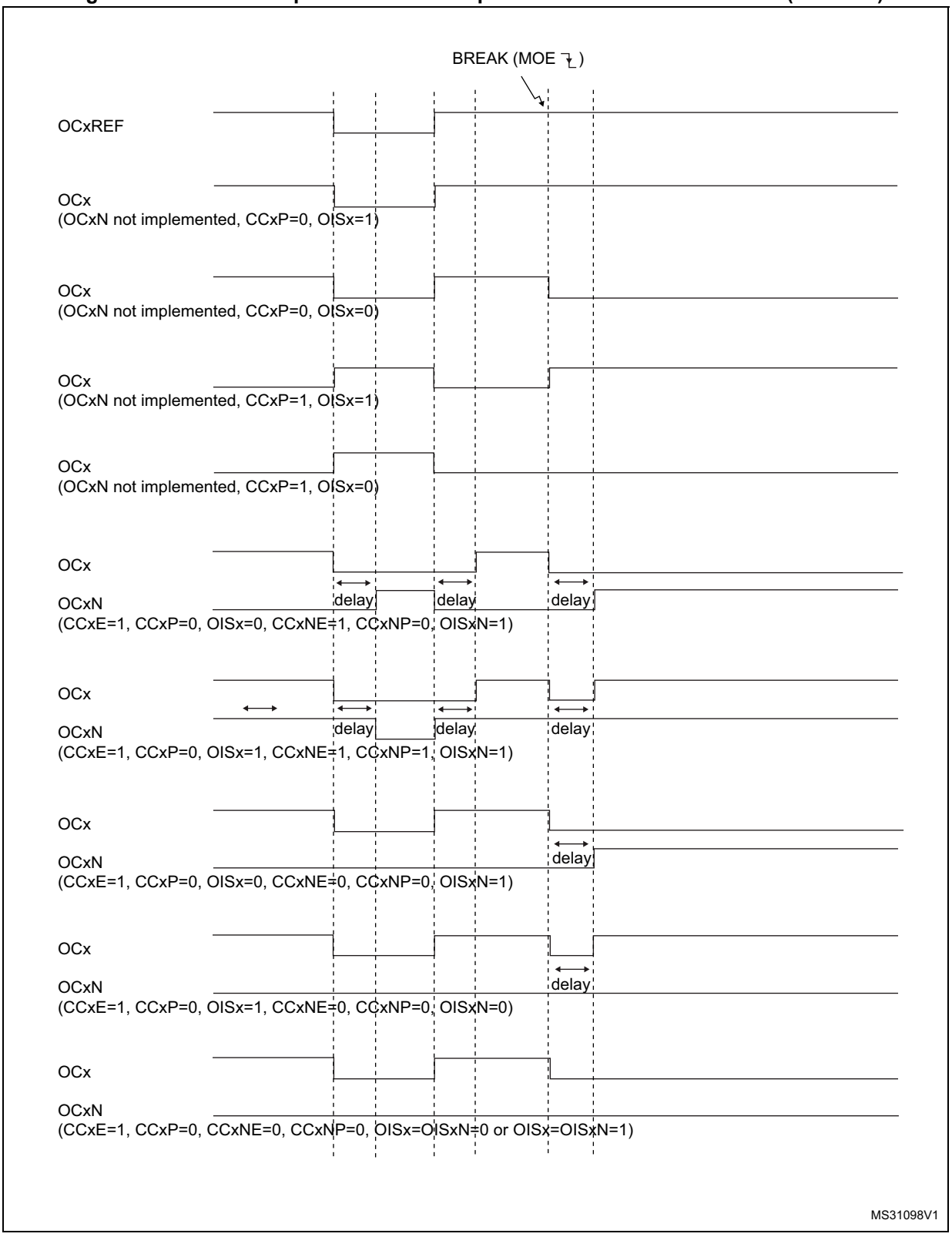
- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to ‘1’ again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 33.4.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)*. The LOCK bits can be written only once after an MCU reset.

Figure 273 shows an example of behavior of the outputs in response to a break.

Figure 273. Various output behavior in response to a break event on BRK (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in *Table 267*.

Note: BRK2 must only be used with OSSR = OSSI = 1.

Table 267. Behavior of timer outputs versus BRK/BRK2 inputs

BRK	BRK2	Timer outputs state	Typical use case	
			OCxN output (low side switches)	OCx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 274 gives an example of OCx and OCxN output behavior in case of active signals on BRK and BRK2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 274. PWM output state following BRK and BRK2 pins assertion (OSSI=1)

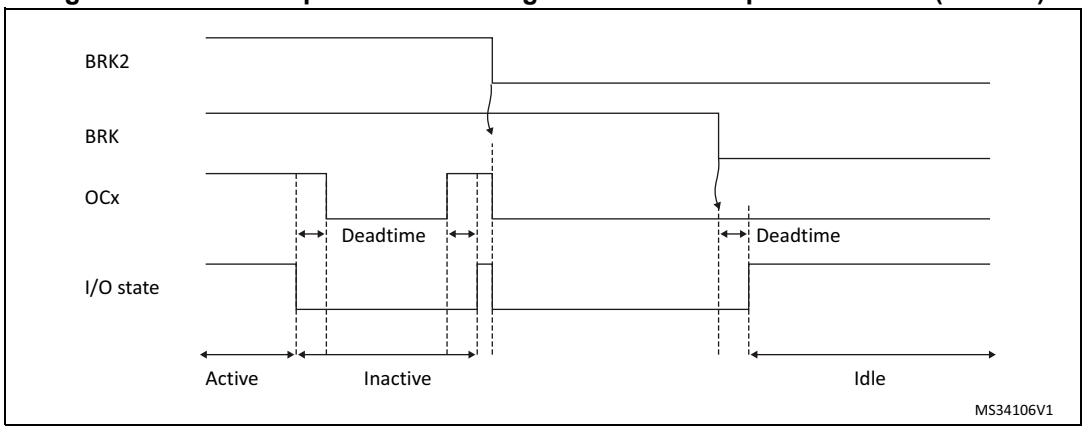
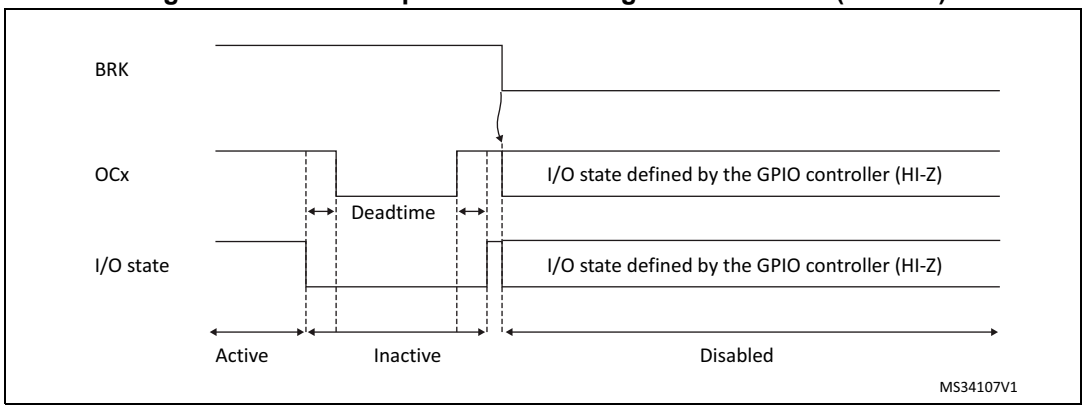


Figure 275. PWM output state following BRK assertion (OSSI=0)



33.3.17 Bidirectional break inputs

The TIM1/TIM8 are featuring bidirectional break I/Os, as represented on *Figure 276*.

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break and break2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the break and break2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BK(2)E = 1). When a software break event is generated with BK(2)E = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break(2) I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BK(2)DSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 268*)

Table 268. Break protection disarming conditions

MOE	BKDIR (BK2DIR)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

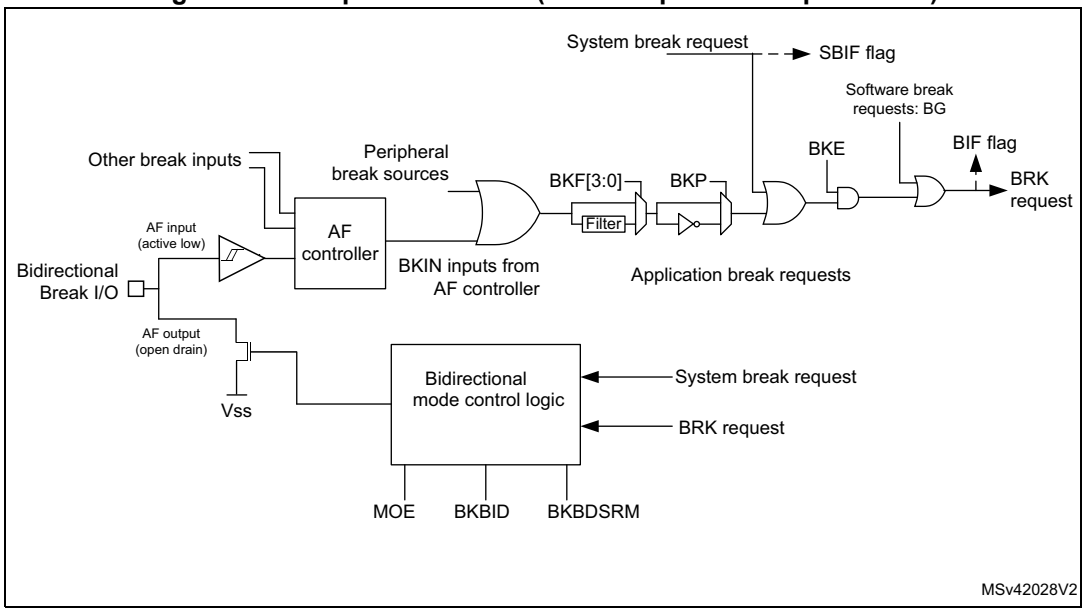
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 276. Output redirection (BRK2 request not represented)



33.3.18 Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next update event (UEV) occurs. This function can be used only in the

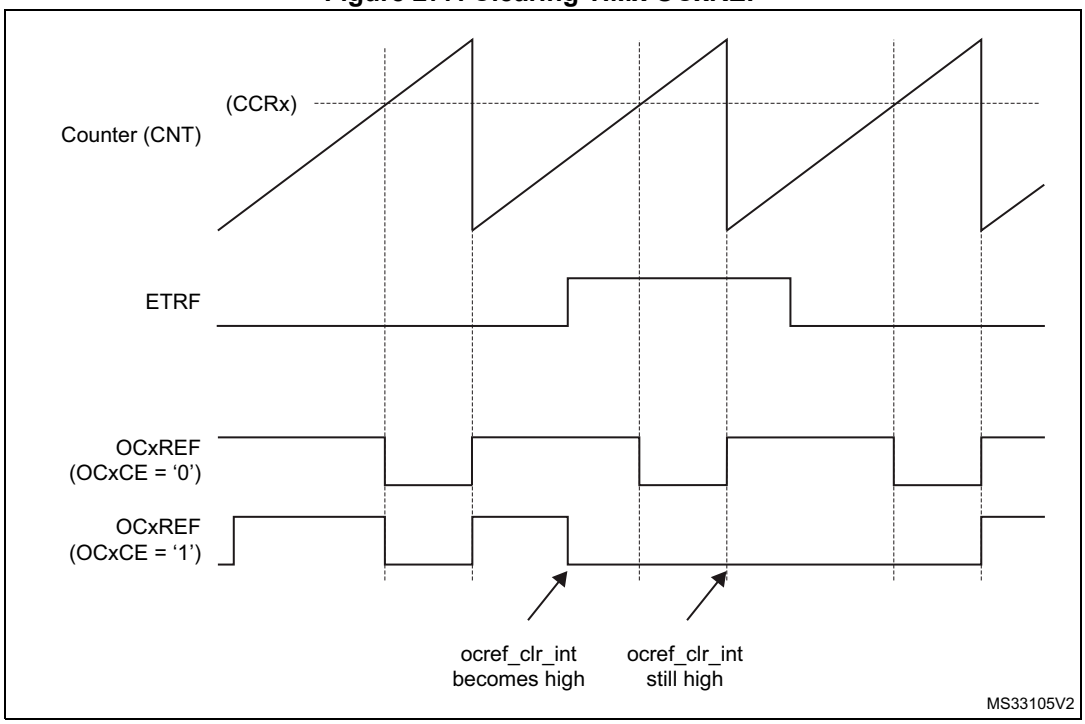
output compare and PWM modes. It does not work in forced mode. For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling.

When ETRF is chosen, ETR must be configured as follows:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

Figure 277 shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 277. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.

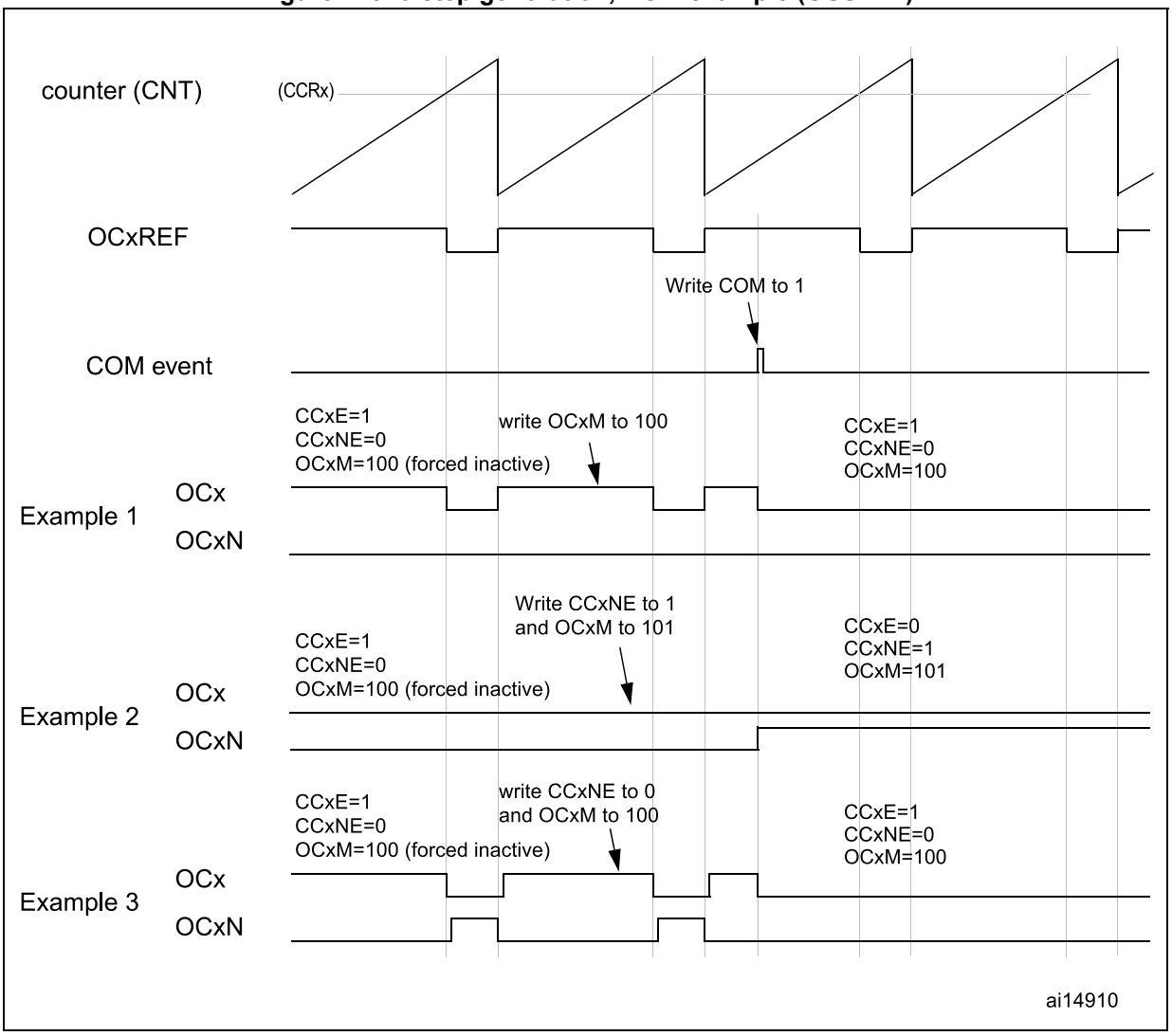
33.3.19 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 278* describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 278. 6-step generation, COM example (OSSR=1)



33.3.20 One-pulse mode

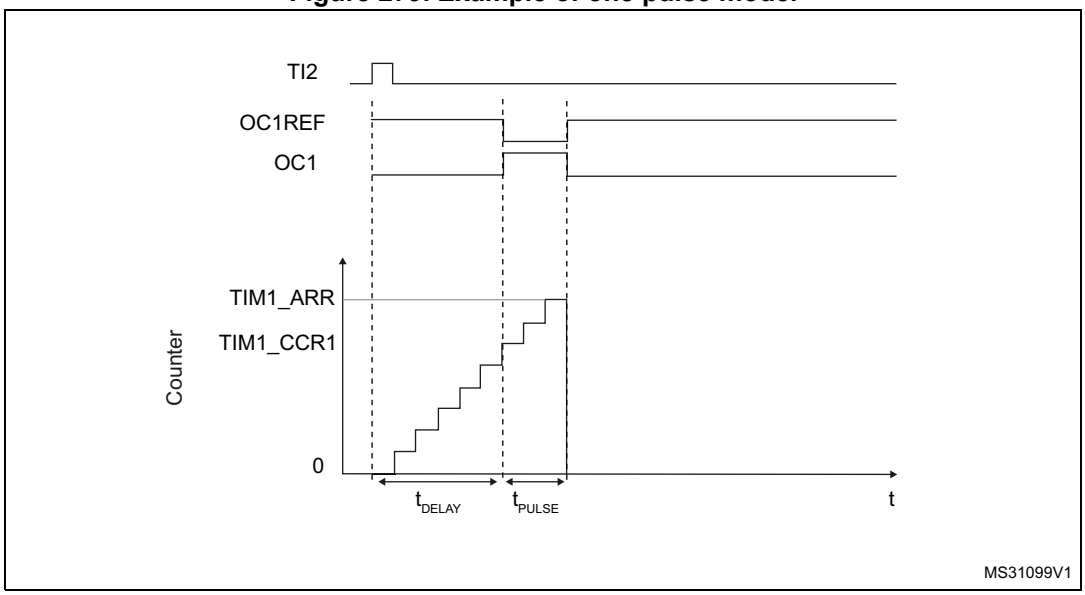
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 279. Example of one pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

33.3.21 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 33.3.20*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

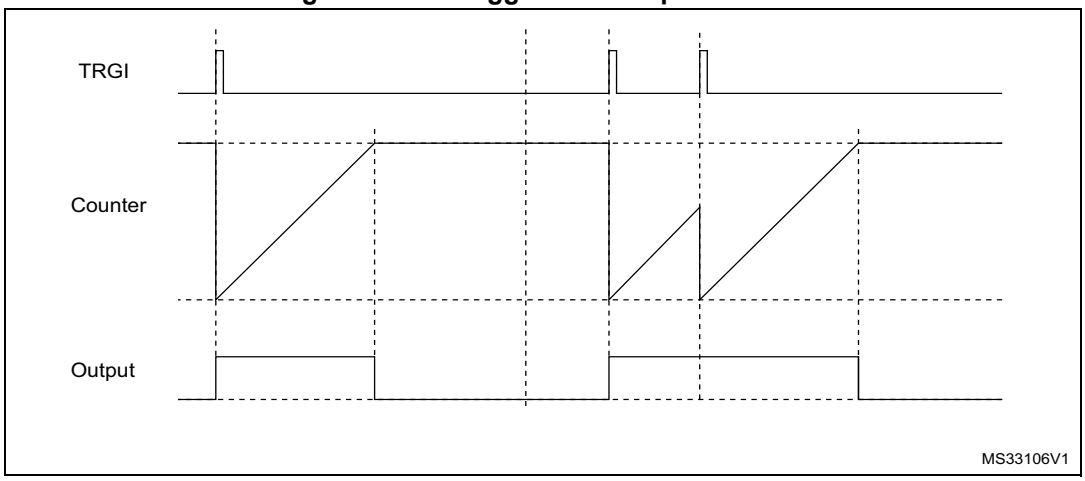
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 280. Retriggerable one pulse mode



33.3.22 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to *Table 269*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Note: The prescaler must be set to zero when encoder mode is enabled

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 269. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The *Figure 281* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 281. Example of counter operation in encoder interface mode.

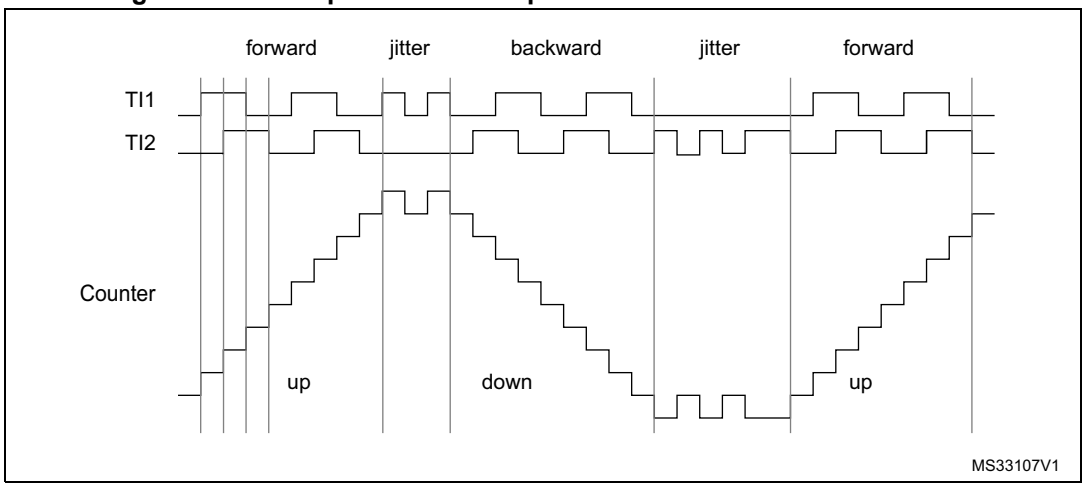
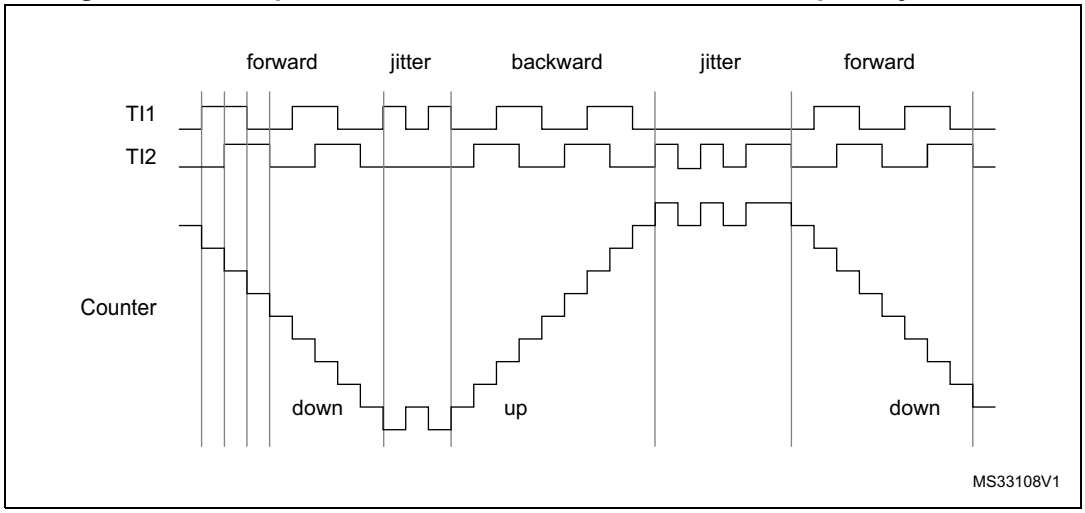


Figure 282 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 282. Example of encoder interface mode with TI1FP1 polarity inverted.



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

33.3.23 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

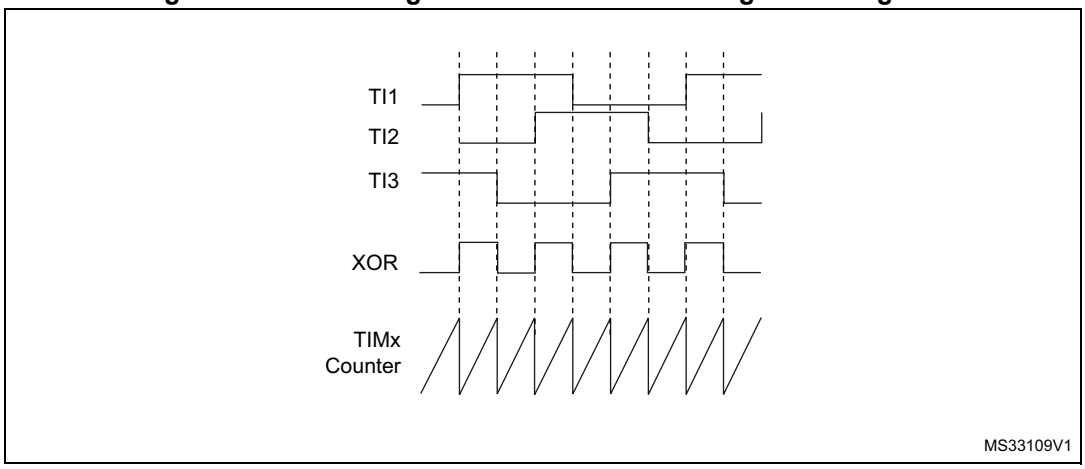
There is no latency between the UIF and UIFCPY flags assertion.

33.3.24 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 283* below.

Figure 283. Measuring time interval between edges on 3 signals



33.3.25 Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1 or TIM8) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4) referred to as “interfacing timer” in *Figure 284*. The “interfacing timer” captures the 3 timer input pins (CC1, CC2, CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (See *Figure 257: Capture/compare channel (example: channel 1 input stage) on page 1098*). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1 or TIM8) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1 or TIM8) through the TRGO output.

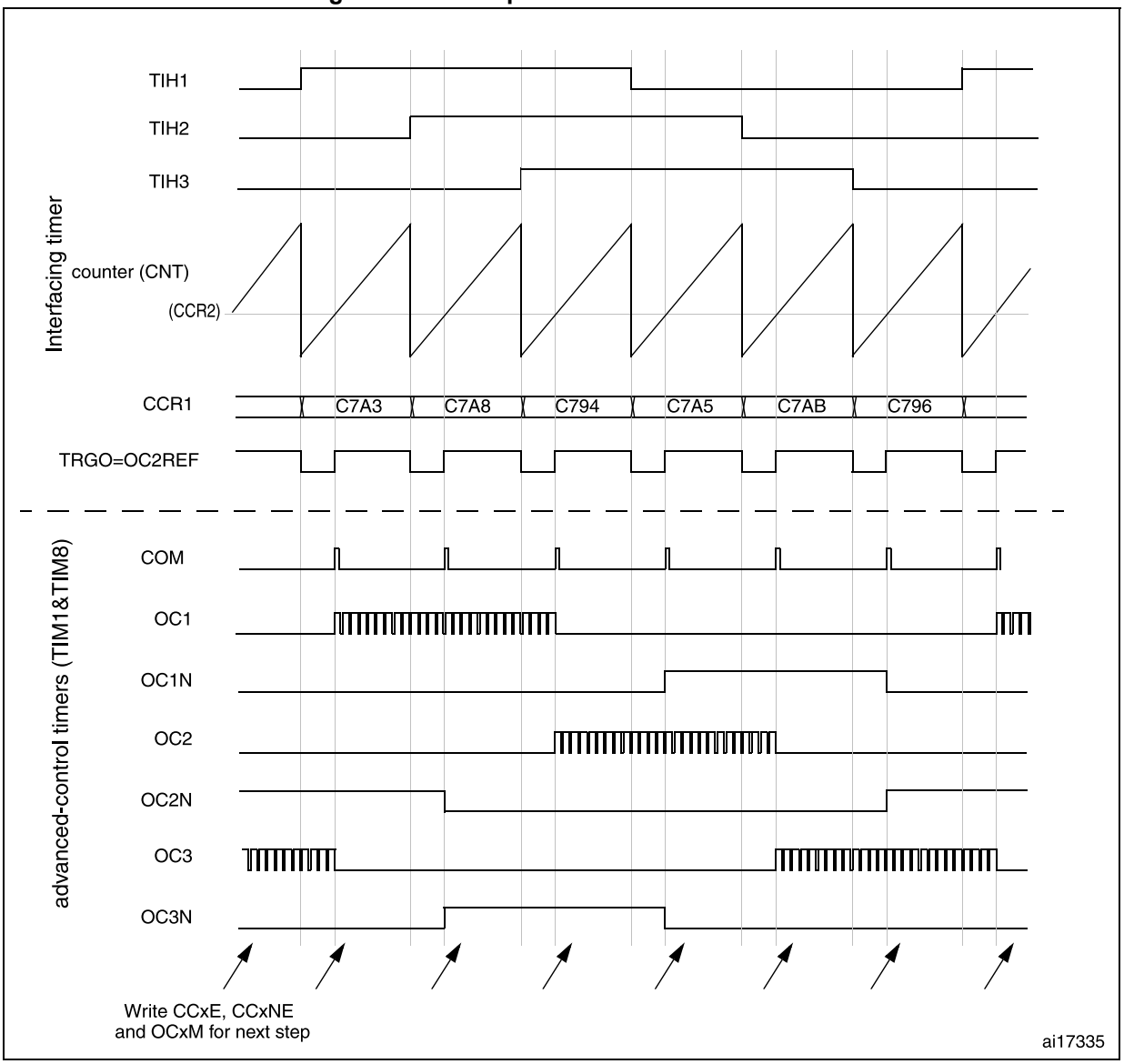
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘01’. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The *Figure 284* describes this example.

Figure 284. Example of Hall sensor interface



33.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 34.3.19: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

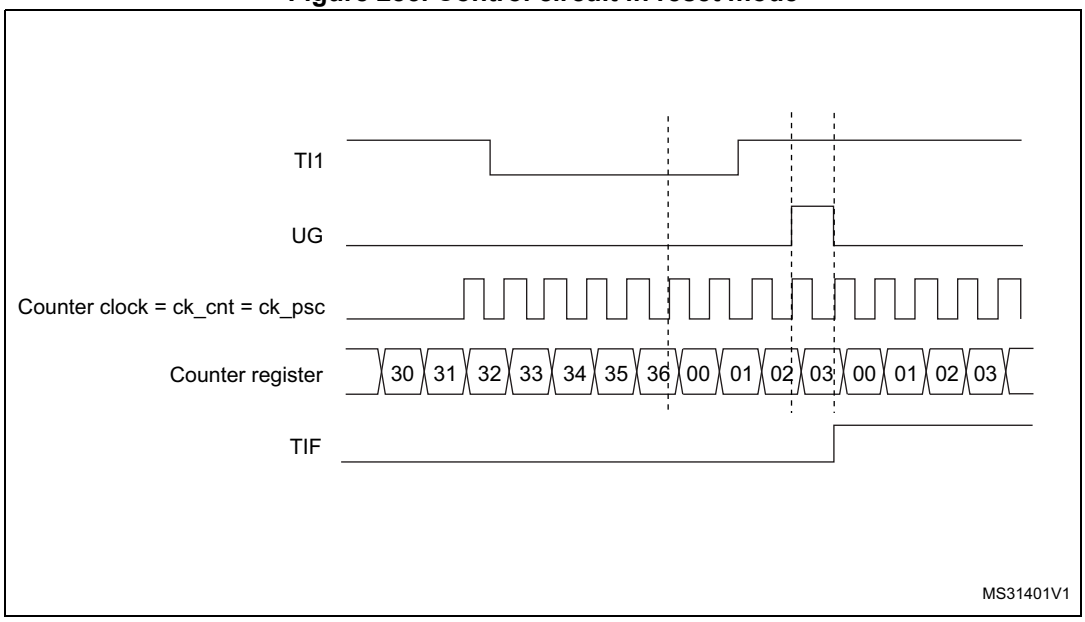
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 285. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

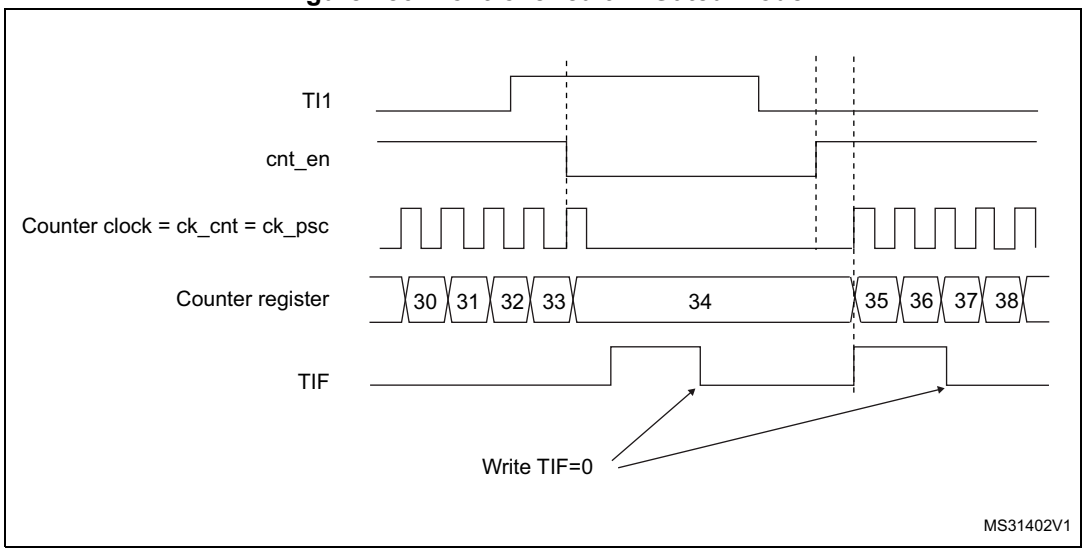
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 286. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1

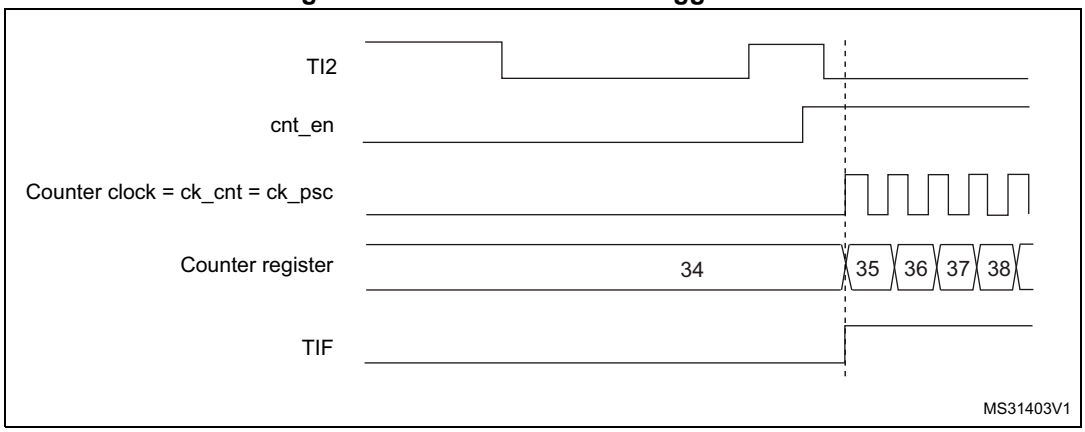
register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 287. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

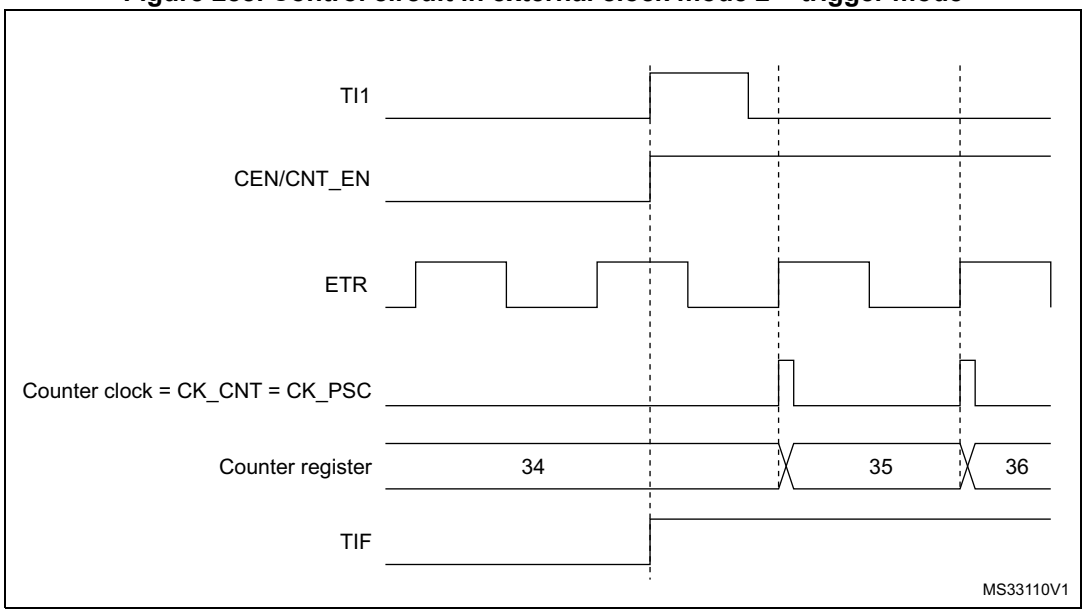
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 288. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

33.3.27 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 268 on page 1110*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

33.3.28 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

33.3.29 Debug mode

When the microcontroller enters debug mode (Cortex®-M33 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in 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).

33.4 TIM1/TIM8 registers

Refer to for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

33.4.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx):
00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Note: $t_{DTS} = 1/f_{DTS}$, $t_{CK\_INT} = 1/f_{CK\_INT}$.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: Switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) is not allowed

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

33.4.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
								rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

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

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).

0100: **Compare** - OC1REFC signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REFC signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REFC signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REFC signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REFC signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REFC signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REFC rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REFC rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REFC or OC6REFC rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REFC rising or OC6REFC falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REFC or OC6REFC rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REFC rising or OC6REFC falling edges generate pulses on TRGO2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output Idle state 6 (OC6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output Idle state 5 (OC5 output)
Refer to OIS1 bit

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)
0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)
0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow selected information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)
Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.

33.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

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.	SMS[3]
															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]			Res.	SMS[2:0]		
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 15 **ETP**: External trigger polarity
This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.
Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of f_{CK_INT} frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

000: Internal Trigger 0 (ITR0)
001: Internal Trigger 1 (ITR1)
010: Internal Trigger 2 (ITR2)
011: Internal Trigger 3 (ITR3)
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)
111: External Trigger input (ETRF)

See *Table 270: TIMx internal trigger connection on page 1142* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Note: The other bit is at position 16 in the same register

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 (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Codes above 1000: Reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 270. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM15	TIM2	TIM3	TIM4
TIM8	TIM1	TIM2	TIM4	TIM5

33.4.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

33.4.5 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF
														rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description (Note: Channel 6 can only be configured as output)

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description (Note: Channel 5 can only be configured as output)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System Break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 33.4.3: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

33.4.6 TIMx event generation register (TIMx_EGR)(x = 1, 8)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

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

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows CCxE, CCxNE and OCxM bits to be updated.

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. The prescaler internal counter is also cleared (the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

33.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter
Refer to IC1F[3:0] description.

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler
Refer to IC1PSC[1:0] description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

33.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output Compare 2 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode
Refer to OC1M[3:0] description.

Bit 11 **OC2PE**: Output Compare 2 preload enable
Refer to OC1PE description.

Bit 10 **OC2FE**: Output Compare 2 fast enable
Refer to OC1FE description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE:** Output Compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

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, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

33.4.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter
Refer to IC1F[3:0] description.

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
Refer to IC1PSC[1:0] description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter
Refer to IC1F[3:0] description.

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler
Refer to IC1PSC[1:0] description.

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

33.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Output compare mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] description.

Bit 11 **OC4PE**: Output compare 4 preload enable
Refer to OC1PE description.

Bit 10 **OC4FE**: Output compare 4 fast enable
Refer to OC1FE description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable
Refer to OC1CE description.

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M[3:0] description.

Bit 3 **OC3PE**: Output compare 3 preload enable
Refer to OC1PE description.

Bit 2 **OC3FE**: Output compare 3 fast enable
Refer to OC1FE description.

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

33.4.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/Compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: OC1N active high.
1: OC1N active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/Compare 1 output polarity
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: The configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 271* 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 271. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states(1)	
MOE bit	**OSSI bit**	**OSSR bit**	**CCxE bit**	**CCxNE bit**	**OCx output state**	**OCxN output state**
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) OCx=0, OCxN=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN = OCxREF xor CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN = OCxREF x or CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Off-State (output enabled with inactive state) OCxN=CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered). Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration). **Note**: BRK2 can only be used if OSSI = OSSR = 1.	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

33.4.12 TIMx counter (TIMx_CNT)(x = 1, 8)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

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

Bits 15:0 **CNT[15:0]**: Counter value

33.4.13 TIMx prescaler (TIMx_PSC)(x = 1, 8)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

33.4.14 TIMx auto-reload register (TIMx_ARR)(x = 1, 8)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 33.3.1: Time-base unit on page 1078* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

33.4.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:0 **REP[15:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

the number of PWM periods in edge-aligned mode

the number of half PWM period in center-aligned mode.

33.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

33.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

33.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

33.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

33.4.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional

Refer to BKBID description

Bit 28 **BKBID**: Break Bidirectional

0: Break input BRK in input mode
1: Break input BRK in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 Disarm

Refer to BKDSRM description

Bit 26 **BKDSRM**: Break Disarm

0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity

0: Break input BRK2 is active low
1: Break input BRK2 is active high

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable

This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 272: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled

Note: The BKIN2 must only be used with OSSR = OSSI = 1.

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK2 acts asynchronously
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).

See OC/OCN enable description for more details (*Section 33.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 272: 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 33.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

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 33.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and B*K2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0]* bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0] x t_{DTG} with $t_{DTG} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x t_{DTG} with $t_{DTG} = 2 \times t_{DTS}$.
DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t_{DTG} with $t_{DTG} = 8 \times t_{DTS}$.
DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t_{DTG} with $t_{DTG} = 16 \times t_{DTS}$.
Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

33.4.21 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
10001: 18 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIMx_CR1.

– If DBL = 7 bytes and DBA = TIMx_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

– If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.

– If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

33.4.22 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

33.4.23 TIM1 option register 1 (TIM1_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP	Res.	Res.	ETR_ADC1_RMP[1:0]	
											rw			rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TI1_RMP**: Input Capture 1 remap

0: TIM1 input capture 1 is connected to I/O

1: TIM1 input capture 1 is connected to COMP1 output.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **ETR_ADC1_RMP[1:0]**: External trigger remap on ADC1 analog watchdog

00 : TIM1_ETR is not connected to ADC1 AWDx. This configuration must be selected when the ETR comes from the I/O.

01 : TIM1_ETR is connected to ADC1 AWD1.

10 : TIM1_ETR is connected to ADC1 AWD2.

11 : TIM1_ETR is connected to ADC1 AWD3.

Note: ADC1 AWDx sources are 'ORed' with the TIM1_ETR input signals. When ADC1 AWDx is used, it is necessary to make sure that the corresponding TIM1_ETR input pin is not enabled in the alternate function controller. Refer to Figure 250: TIM1 ETR input circuitry.

33.4.24 TIM8 option register 1 (TIM8_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP	Res.	Res.	Res.	Res.
											rw				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TI1_RMP**: Input Capture 1 remap
0: TIM8 input capture 1 is connected to I/O
1: TIM8 input capture 1 is connected to COMP2 output.

Bits 3:0 Reserved, must be kept at reset value.

33.4.25 TIMx capture/compare mode register 3 (TIMx_CCMR3)(x = 1, 8)

Address offset: 0x54

Reset value: 0x0000 0000

The channels 5 and 6 can only be configured in output.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable
Refer to OC1CE description.

Bits 24, 14, 13, 12 **OC6M[3:0]**: Output compare 6 mode
Refer to OC1M description.

Bit 11 **OC6PE**: Output compare 6 preload enable
Refer to OC1PE description.

Bit 10 **OC6FE**: Output compare 6 fast enable
Refer to OC1FE description.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable
Refer to OC1CE description.

Bits 16, 6, 5, 4 **OC5M[3:0]**: Output compare 5 mode
Refer to OC1M description.

Bit 3 **OC5PE**: Output compare 5 preload enable
Refer to OC1PE description.

Bit 2 **OC5FE**: Output compare 5 fast enable
Refer to OC1FE description.

Bits 1:0 Reserved, must be kept at reset value.

33.4.26 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group Channel 5 and Channel 3

Distortion on Channel 3 output:

0: No effect of OC5REF on OC3REFC

1: OC3REFC is the logical AND of OC3REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group Channel 5 and Channel 2

Distortion on Channel 2 output:

0: No effect of OC5REF on OC2REFC

1: OC2REFC is the logical AND of OC2REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group Channel 5 and Channel 1

Distortion on Channel 1 output:

0: No effect of OC5REF on OC1REFC5

1: OC1REFC is the logical AND of OC1REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value

CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC5 output.

33.4.27 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC6 output.

33.4.28 TIM1 option register 2 (TIM1_OR2)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[2]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK0E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
rw	rw			rw	rw	rw	rw						rw	rw	rw

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

Bits 16:14 **ETRSEL[2:0]**: ETR source selection

These bits select the ETR input source.

000: ETR input is connected to I/O

001: COMP1 output

010: COMP2 output

011: ADC1 AWD1

100: ADC1 AWD2

101: ADC1 AWD3

Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: COMP2 input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: COMP1 input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: BKIN input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK0E**: BRK dfsdm1_break[0] enable

This bit enables the dfsdm1_break[0] for the timer's BRK input. dfsdm1_break[0] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[0] input disabled

1: dfsdm1_break[0] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 250: TIM1 ETR input circuitry and to Figure 272: Break and Break2 circuitry overview.

33.4.29 TIM1 option register 3 (TIM1_OR3)

Address offset: 0x64

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BK2 CMP2 P	BK2 CMP1 P	BK2 INP	BK2DF1 BK1E	Res.	Res.	Res.	Res.	Res.	BK2 CMP2E	BK2 CMP1E	BK2INE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: COMP2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP1 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: COMP1 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: BKIN2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2DF1BK1E**: BRK2 dfsdm1_break[1] enable

This bit enables the dfsdm1_break[1] for the timer's BRK2 input. dfsdm1_break[1] output is 'ORed' with the other BRK2 sources.

0: dfsdm1_break[1] input disabled

1: dfsdm1_break[1] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BK2CMP2E**: BRK2 COMP2 enable

This bit enables the COMP2 for the timer's BRK2 input. COMP2 output is 'ORed' with the other BRK2 sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: BRK2 COMP1 enable

This bit enables the COMP1 for the timer's BRK2 input. COMP1 output is 'ORed' with the other BRK2 sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: BRK2 BKIN input enable

This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.

0: BKIN2 input disabled

1: BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 272: Break and Break2 circuitry overview.

33.4.30 TIM8 option register 2 (TIM8_OR2)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	BK CMP2 P	BK CMP1 P	BKINP	BKDF1 BK2E	Res.	Res.	Res.	Res.	Res.	BK CMP2E	BK CMP1E	BKINE
rw	rw			rw	rw	rw	rw						rw	rw	rw

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

Bits 16:14 **ETRSEL[2:0]**: ETR source selection
These bits select the ETR input source.
000: ETR input is connected to I/O
001: COMP1 output
010: COMP2 output
Others: Reserved
Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input polarity is not inverted (active low if BKP=0, active high if BKP=1)
1: COMP2 input polarity is inverted (active high if BKP=0, active low if BKP=1)
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input polarity is not inverted (active low if BKP=0, active high if BKP=1)
1: COMP1 input polarity is inverted (active high if BKP=0, active low if BKP=1)
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input polarity is not inverted (active low if BKP=0, active high if BKP=1)
1: BKIN input polarity is inverted (active high if BKP=0, active low if BKP=1)
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK2E**: BRK dfsdm1_break[2] enable
This bit enables the dfsdm1_break[2] for the timer's BRK input. dfsdm1_break[2] output is 'ORed' with the other BRK sources.
0: dfsdm1_break[2] input disabled
1: dfsdm1_break[2] input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BKCMP2E**: BRK COMP2 enable
This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.
0: COMP2 input disabled
1: COMP2 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 251: TIM8 ETR input circuitry and to Figure 272: Break and Break2 circuitry overview.

33.4.31 TIM8 option register 3 (TIM8_OR3)

Address offset: 0x64

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BK2 CMP2 P	BK2 CMP1 P	BK2 INP	BK2DF1 BK3E	Res.	Res.	Res.	Res.	Res.	BK2 CMP2E	BK2 CMP1E	BK2INE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: COMP2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP1 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: COMP1 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: BKIN2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2DF1BK3E**: BRK2 dfsdm1_break[3] enable

This bit enables the dfsdm1_break[3] for the timer's BRK2 input. dfsdm1_break[3] output is 'ORed' with the other BRK2 sources.

0: dfsdm1_break[3] input disabled

1: dfsdm1_break[3] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BK2CMP2E**: BRK2 COMP2 enable

This bit enables the COMP2 for the timer's BRK2 input. COMP2 output is 'ORed' with the other BRK2 sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: BRK2 COMP1 enable

This bit enables the COMP1 for the timer's BRK2 input. COMP1 output is 'ORed' with the other BRK2 sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: BRK2 BKIN input enable

This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.

0: BKIN2 input disabled

1: BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 272: Break and Break2 circuitry overview.

33.4.32 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 272. TIM1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM1_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM1_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	**TIM1_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x0C	**TIM1_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**TIM1_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value															0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**TIM1_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	**TIM1_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIM1_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM1_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 272. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**TIM1_CNT**	UIFCP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM1_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM1_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM1_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**TIM1_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM1_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIM1_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIM1_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIM1_BDTR**	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIM1_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIM1_DMAR**	DMAB[31:0]																															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM1_OR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP	Res.	Res.	ETR_ADC1_RMP	
	Reset value																												0			0	0
0x54	**TIM1_CCMR3** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		

Table 272. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x58	**TIM1_CCR5**	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]															
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	**TIM1_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**TIM1_OR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2:0]			Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK0E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																0	0	0			0	0	0	0					0	0	1
0x64	**TIM1_OR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP2P	BK2CMP1P	BK2INP	BK2DF1BK1E	Res.	Res.	Res.	Res.	Res.	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value																					0	0	0	0						0	0	1

Refer to *Section 2.3 on page 87* for the register boundary addresses.

33.4.33 TIM8 register map

TIM8 registers are mapped as 16-bit addressable registers as described in the table below:

Table 273. TIM8 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	**TIM8_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM8_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	**TIM8_SMCR**	Res.	Res.	Res	Res	Res	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	SMS[3]	ETP	ECE	ETP S [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
0x0C	**TIM8_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**TIM8_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value															0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 273. TIM8 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	**TIM8_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	**TIM8_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
	TIM8_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIM8_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM8_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM8_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**TIM8_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
0x28	**TIM8_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	**TIM8_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM8_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM8_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIM8_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIM8_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 273. TIM8 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
0x44	**TIM8_BDTR**	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIM8_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIM8_DMAR**	DMAB[31:0]																															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	**TIM8_CCMR3** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x58	**TIM8_CCR5**	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]															
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	**TIM8_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**TIM8_OR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2:0]			Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																0	0	0			0	0	0	0						0	0	1
0x64	**TIM8_OR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP2P	BK2CMP1P	BK2INP	BK2DF1BK3E	Res.	Res.	Res.	Res.	Res.	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value																					0	0	0	0						0	0	1

Refer to *Section 2.3 on page 87* for the register boundary addresses.

34 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

34.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit/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.

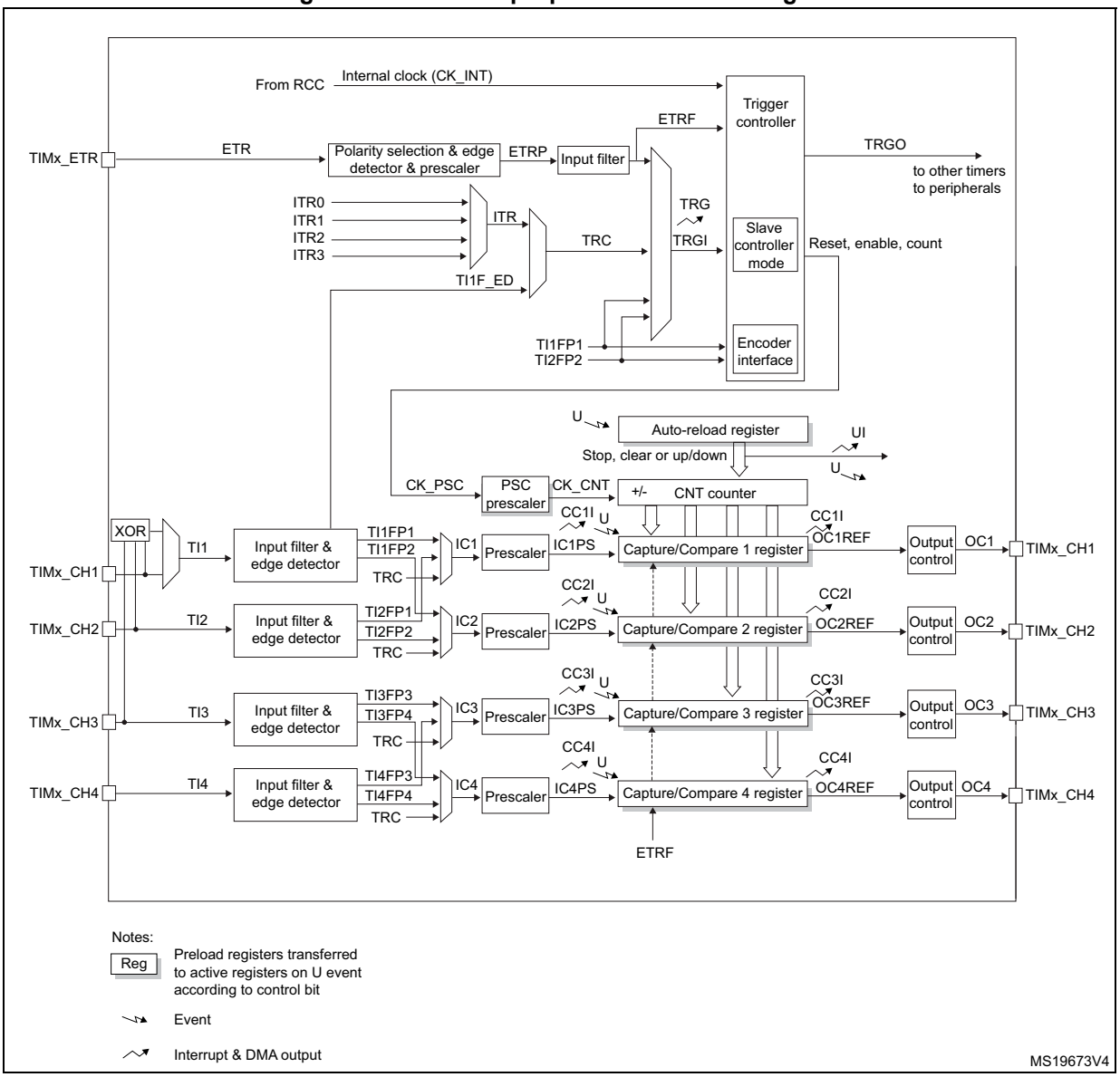
The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 34.3.19: Timer synchronization*.

34.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3, TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 289. General-purpose timer block diagram



34.3 TIM2/TIM3/TIM4/TIM5 functional description

34.3.1 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 but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 290 and *Figure 291* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 290. Counter timing diagram with prescaler division change from 1 to 2

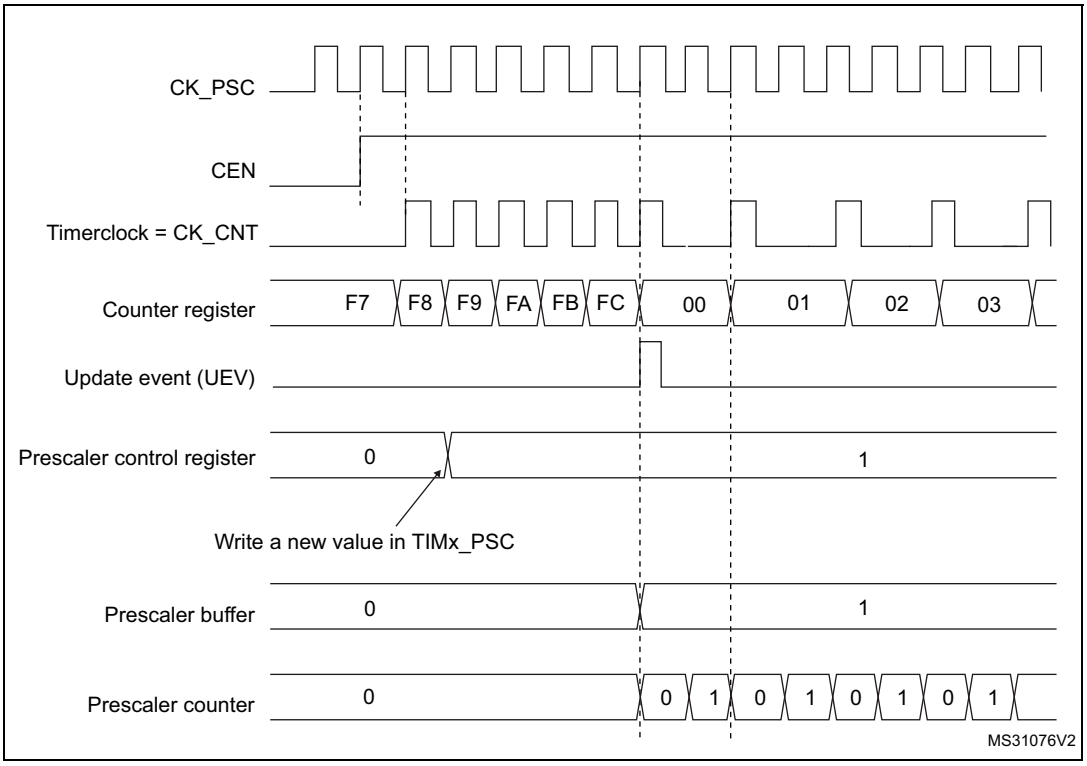
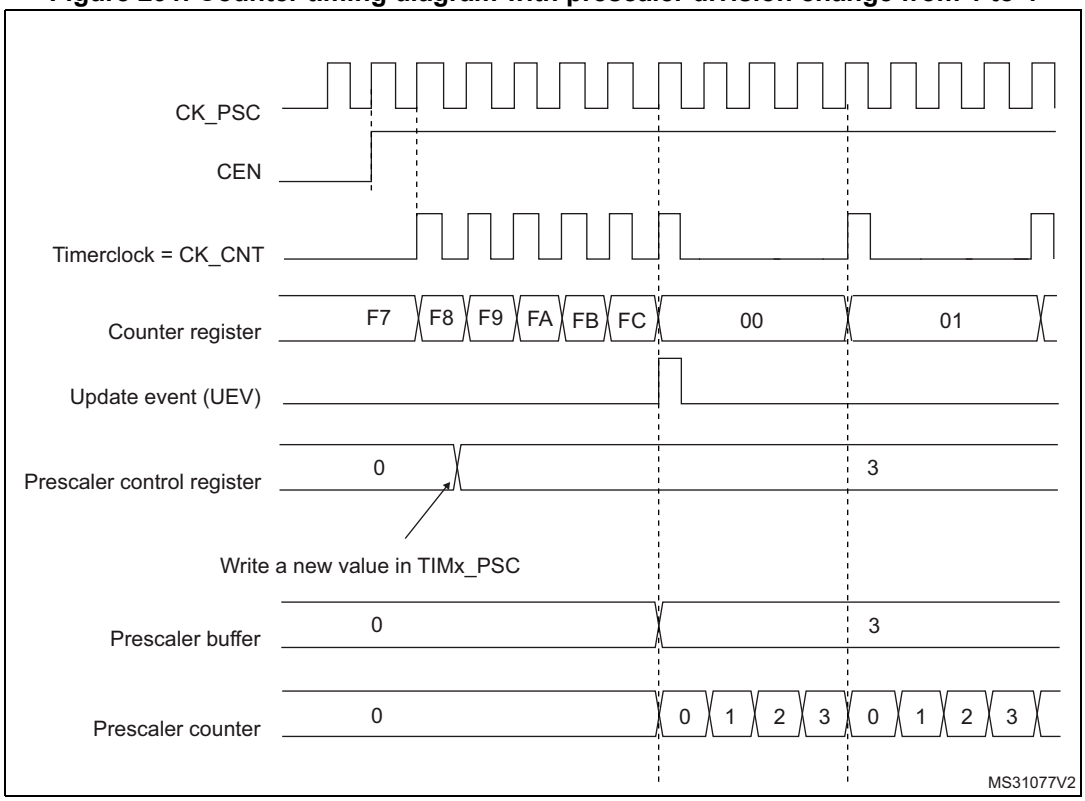


Figure 291. Counter timing diagram with prescaler division change from 1 to 4



34.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 292. Counter timing diagram, internal clock divided by 1

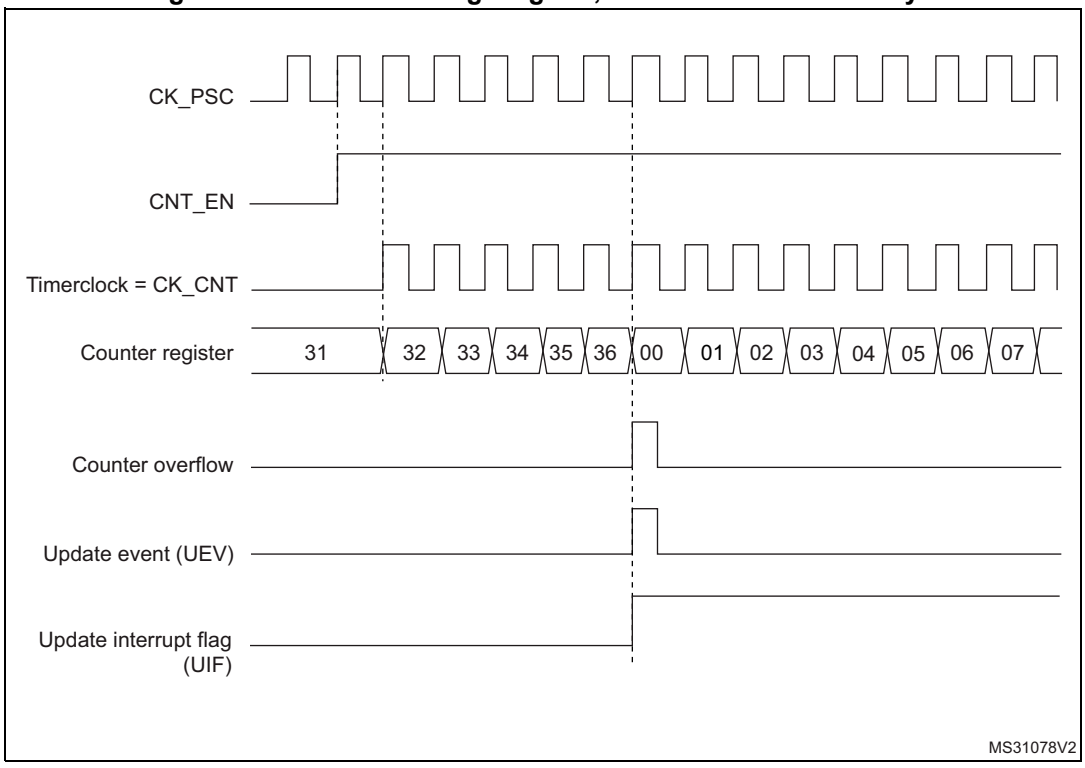


Figure 293. Counter timing diagram, internal clock divided by 2

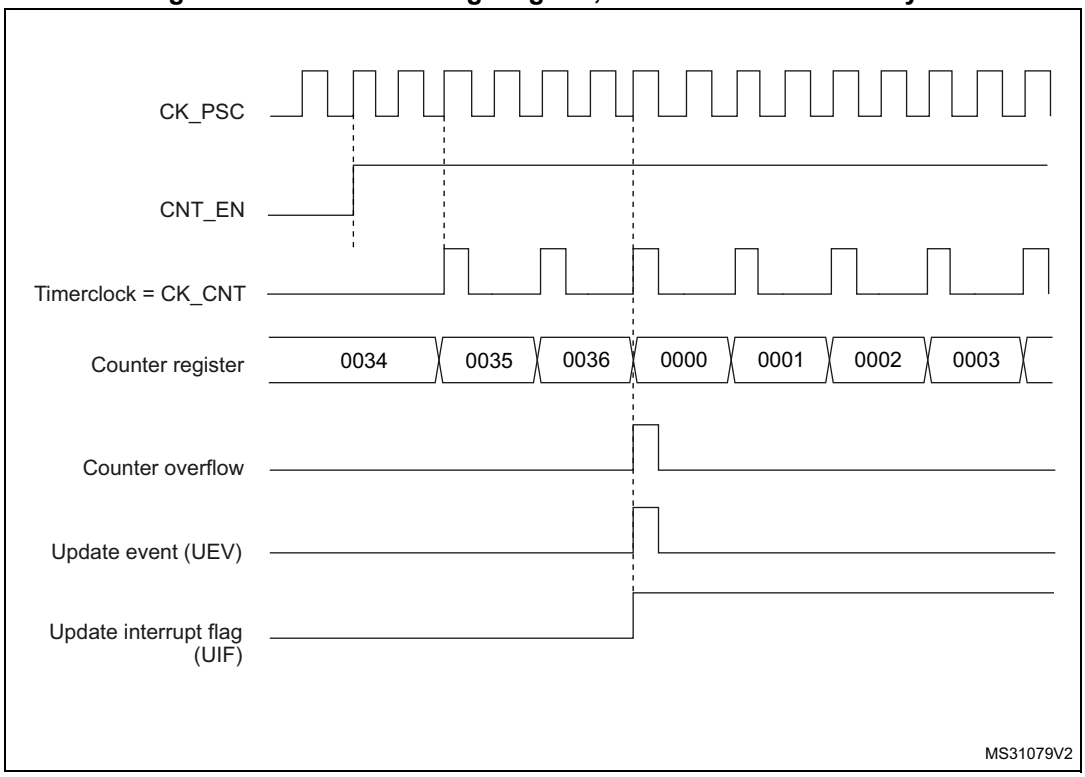


Figure 294. Counter timing diagram, internal clock divided by 4

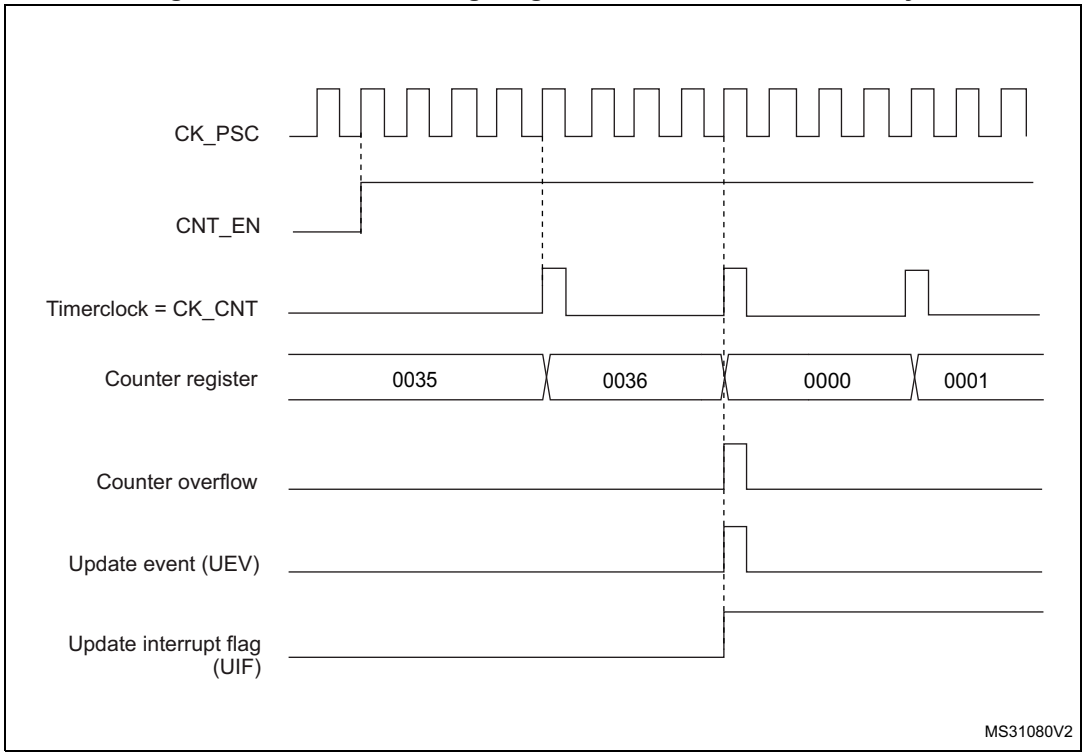


Figure 295. Counter timing diagram, internal clock divided by N

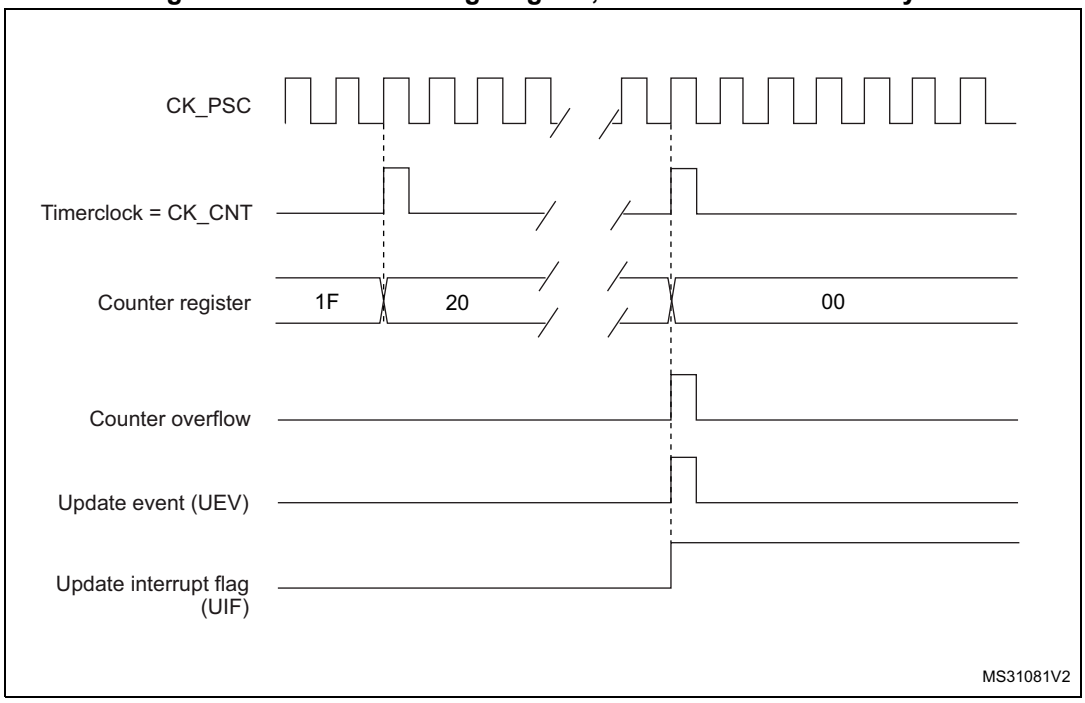


Figure 296. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

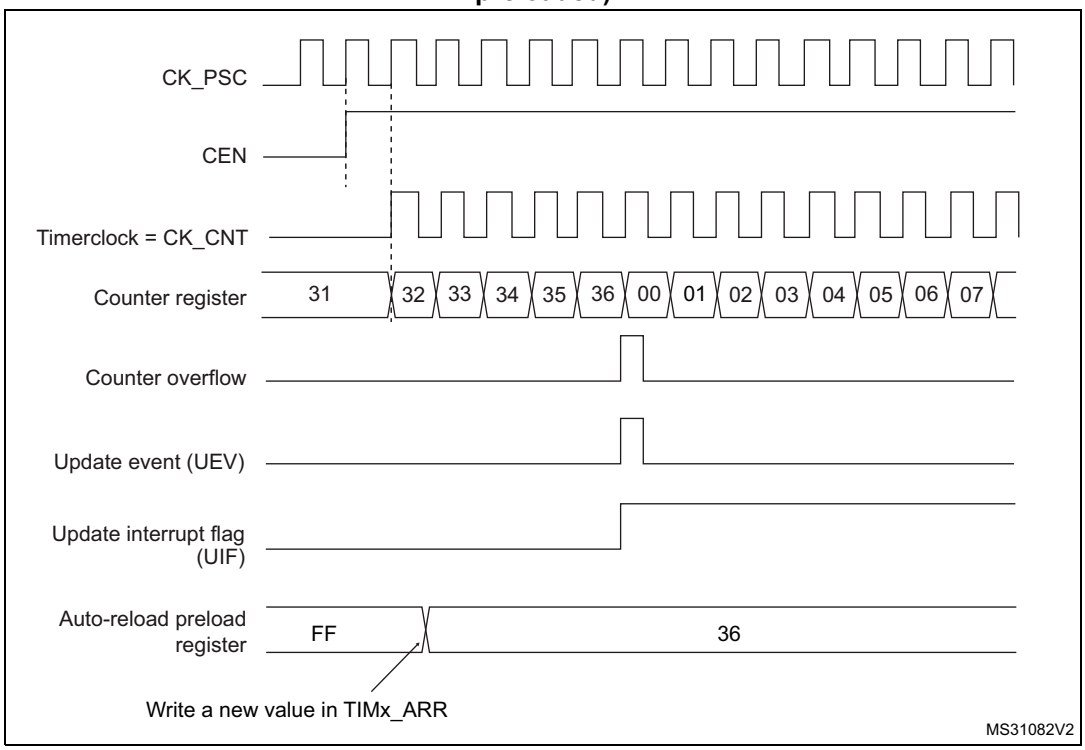
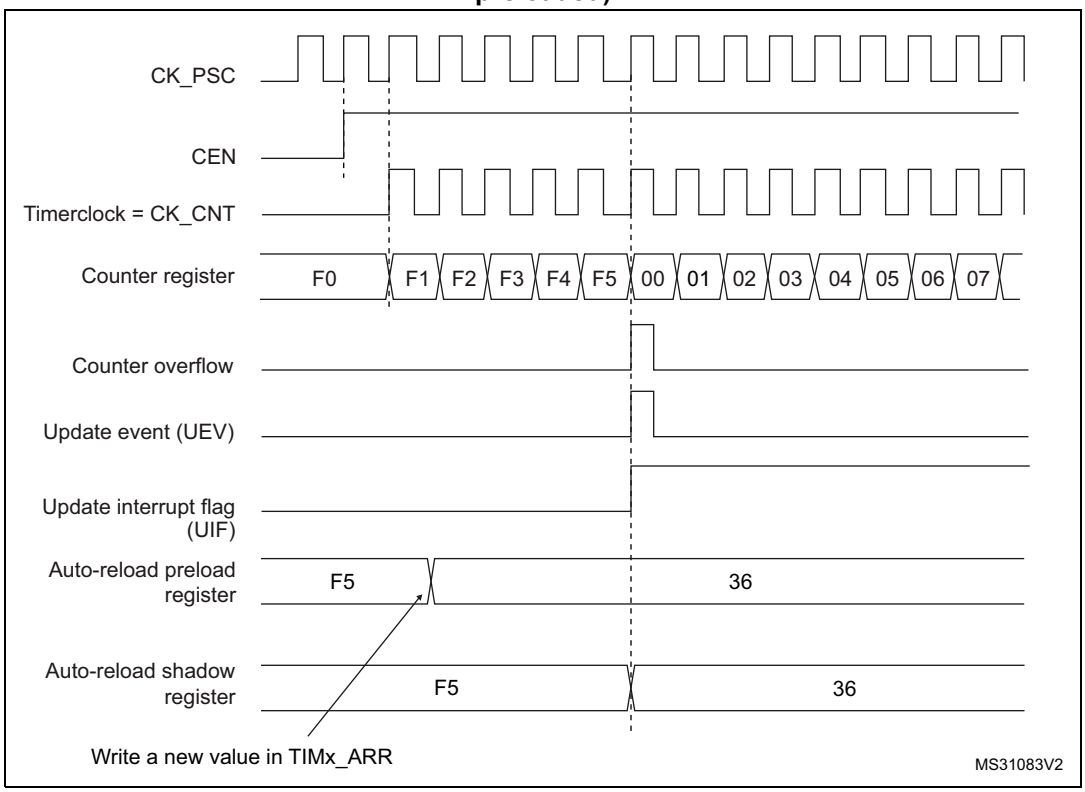


Figure 297. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



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 298. Counter timing diagram, internal clock divided by 1

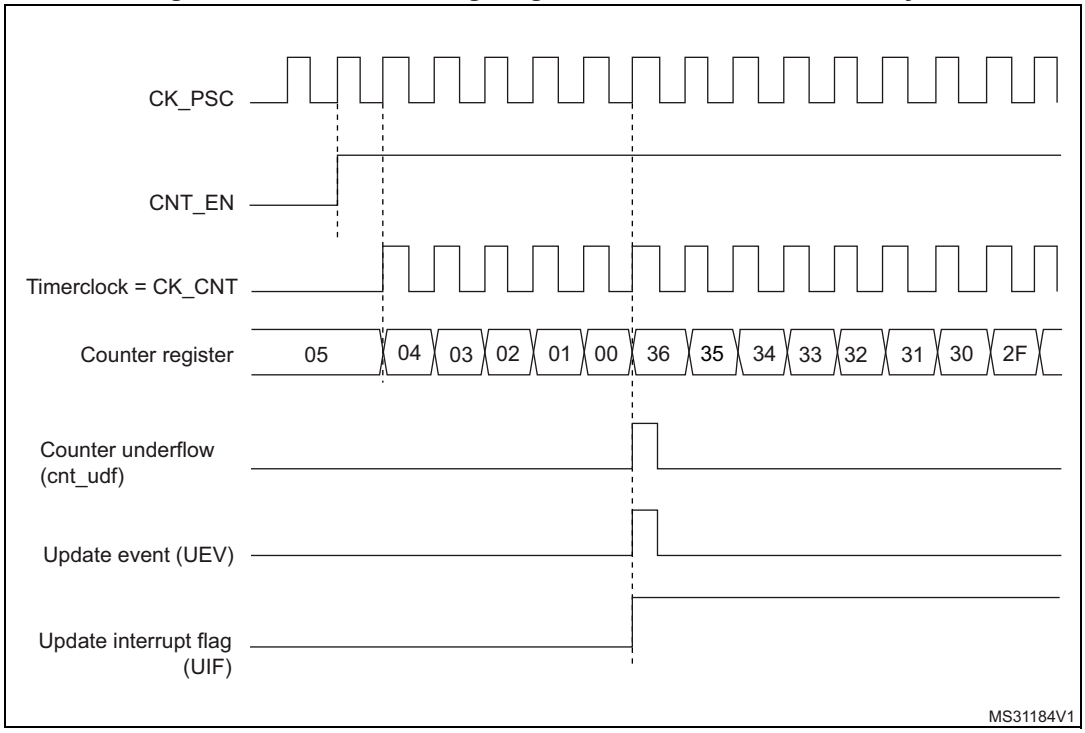


Figure 299. Counter timing diagram, internal clock divided by 2

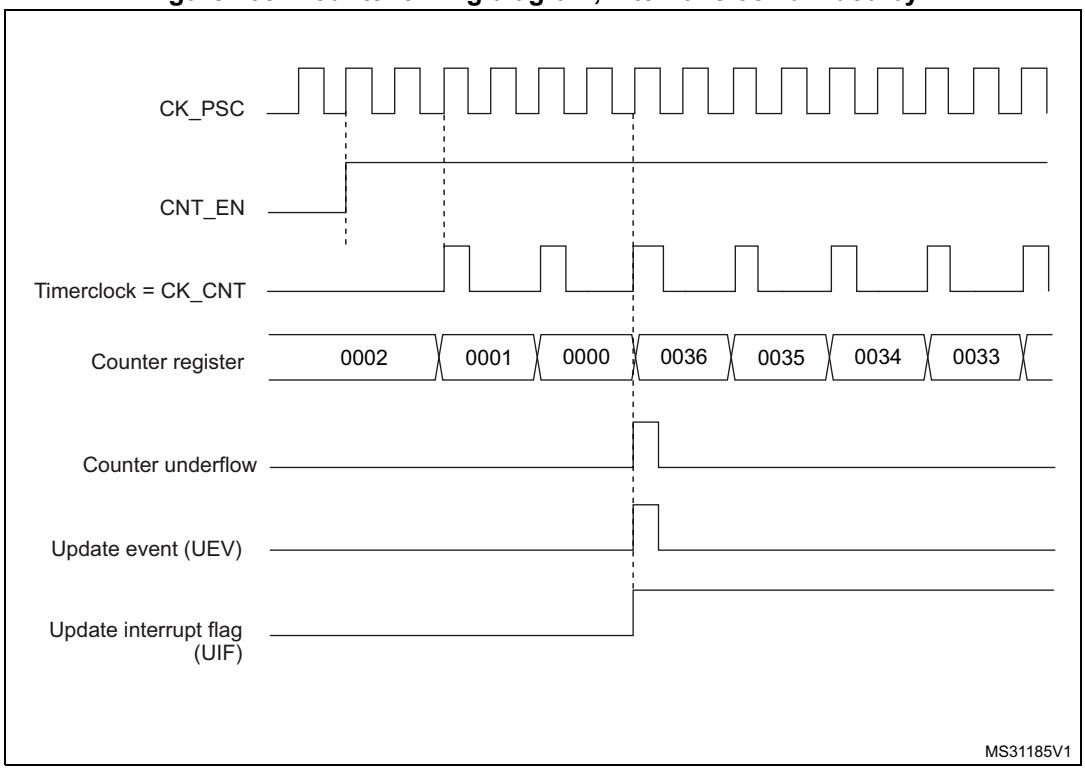


Figure 300. Counter timing diagram, internal clock divided by 4

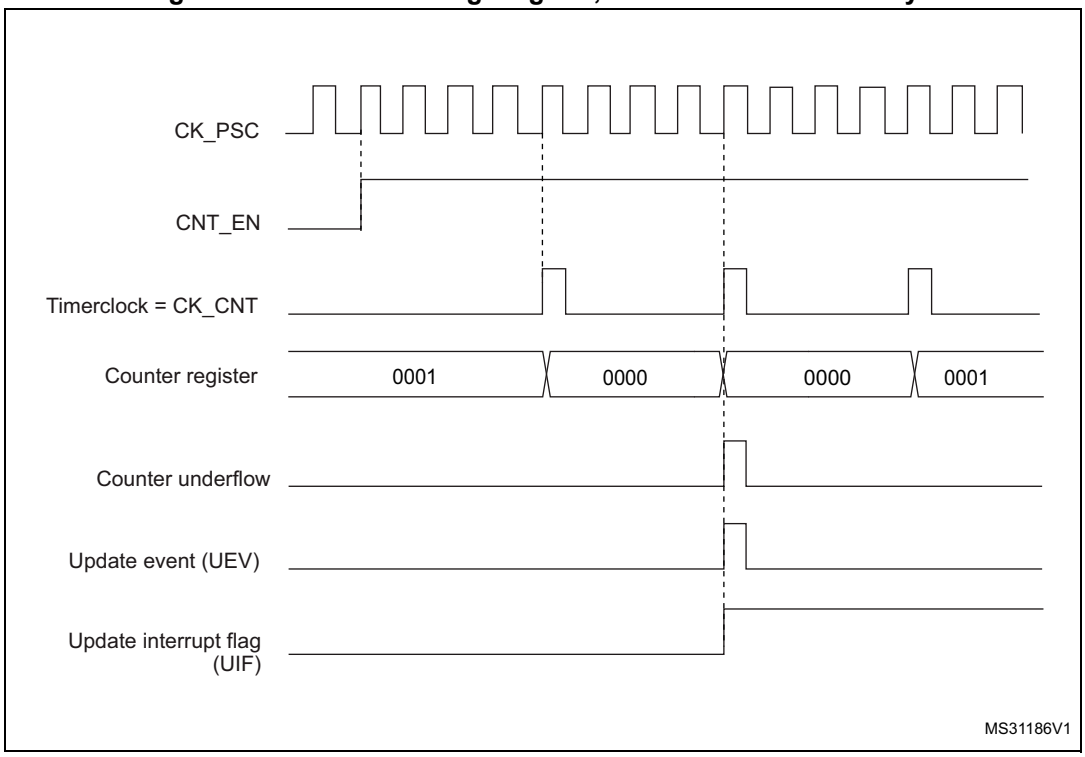


Figure 301. Counter timing diagram, internal clock divided by N

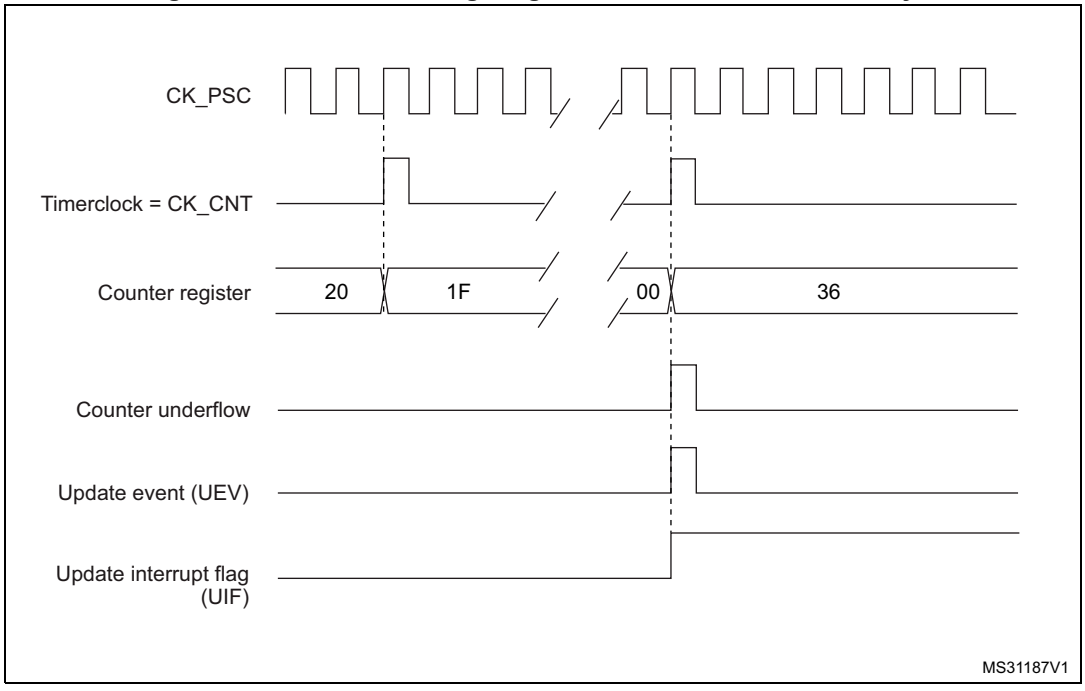
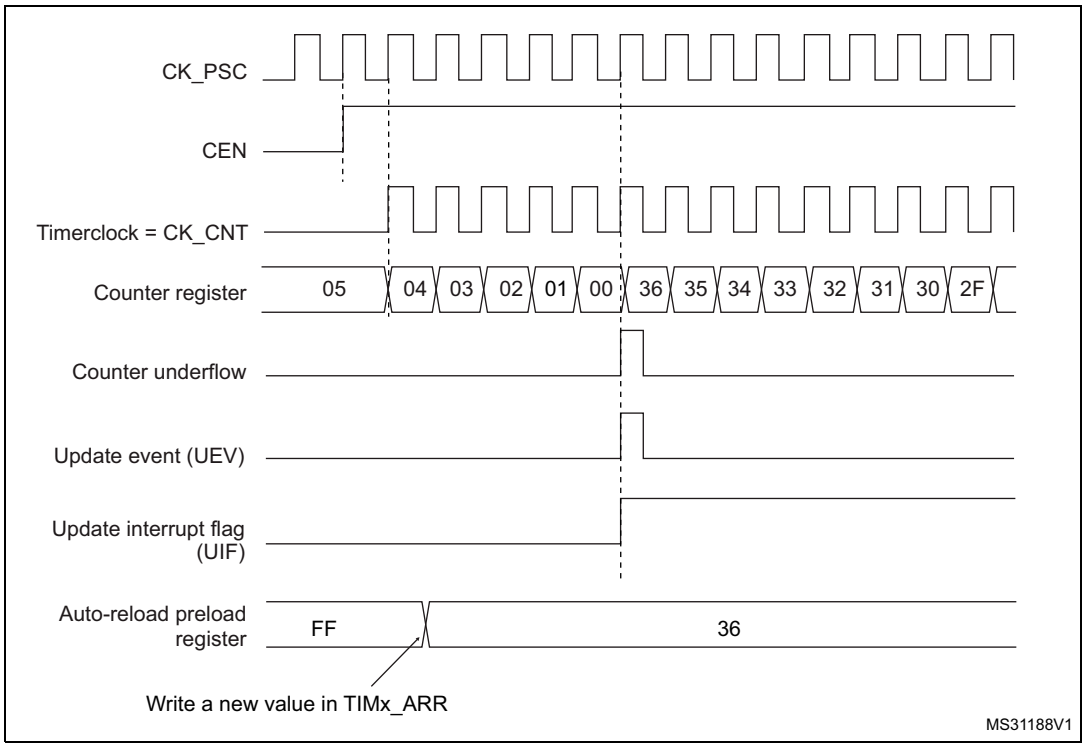


Figure 302. Counter timing diagram, Update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or

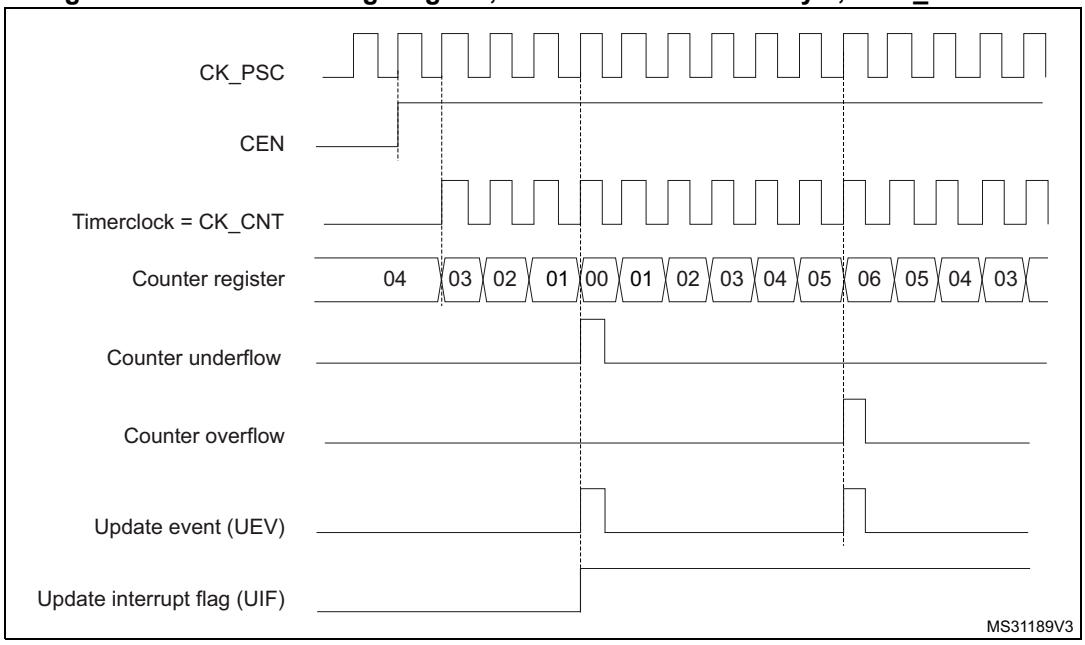
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 303. 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 34.4.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5) on page 1227*).

Figure 304. Counter timing diagram, internal clock divided by 2

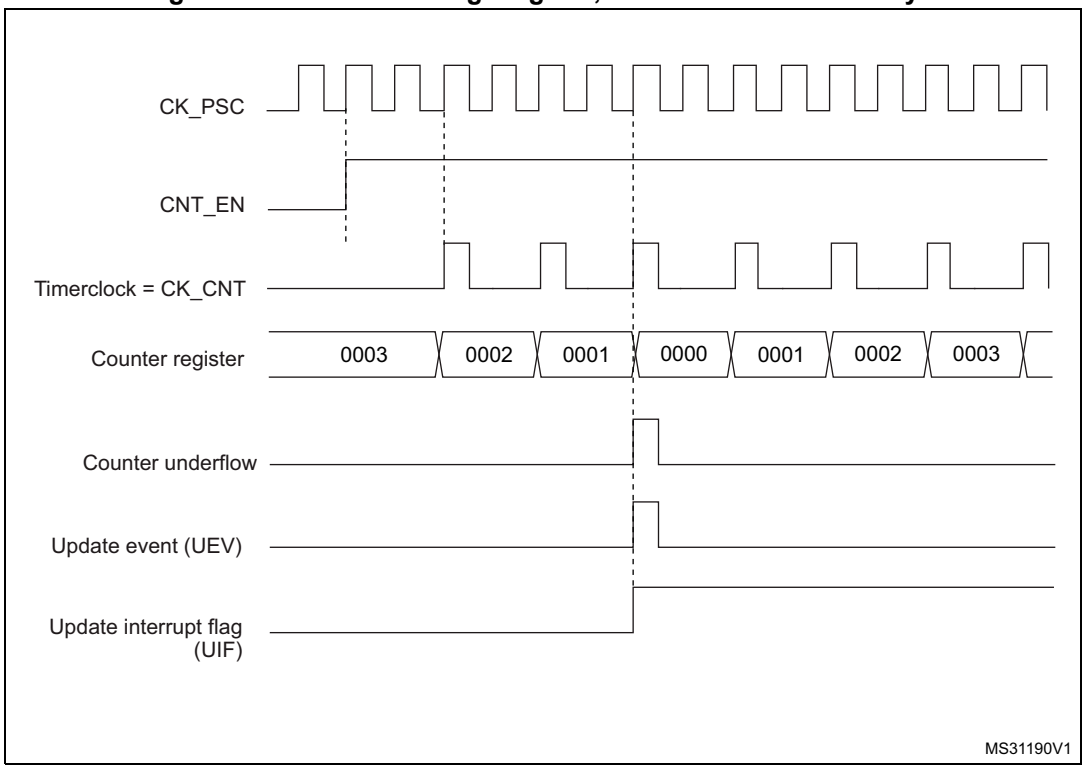
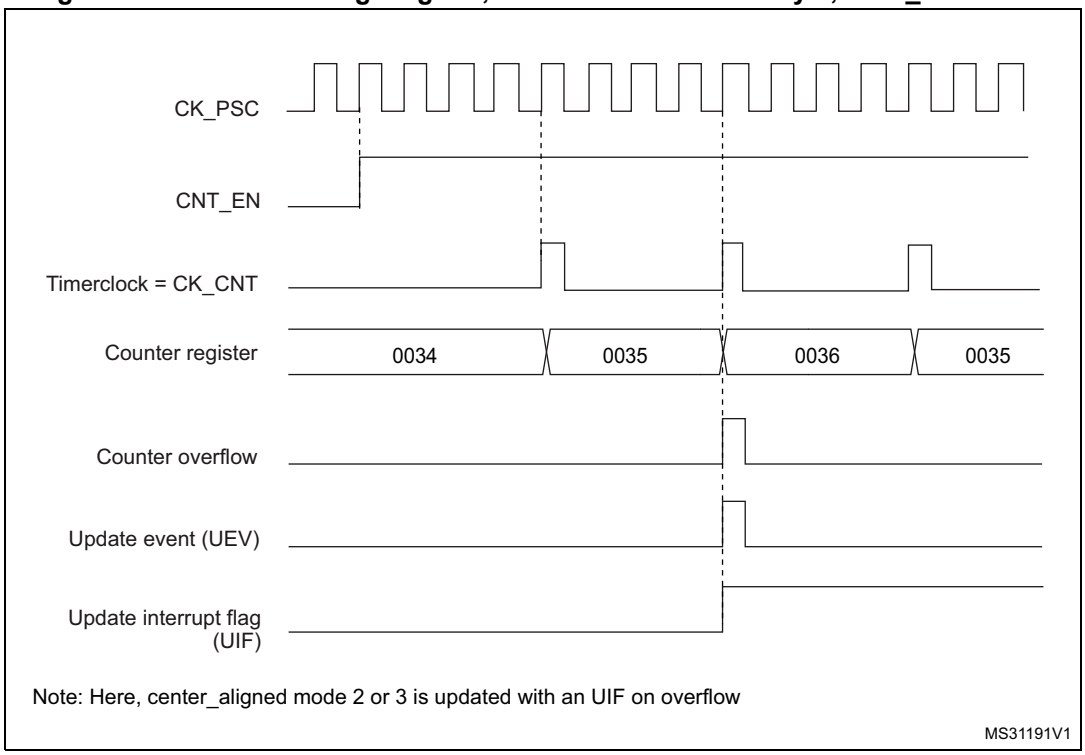


Figure 305. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 306. Counter timing diagram, internal clock divided by N

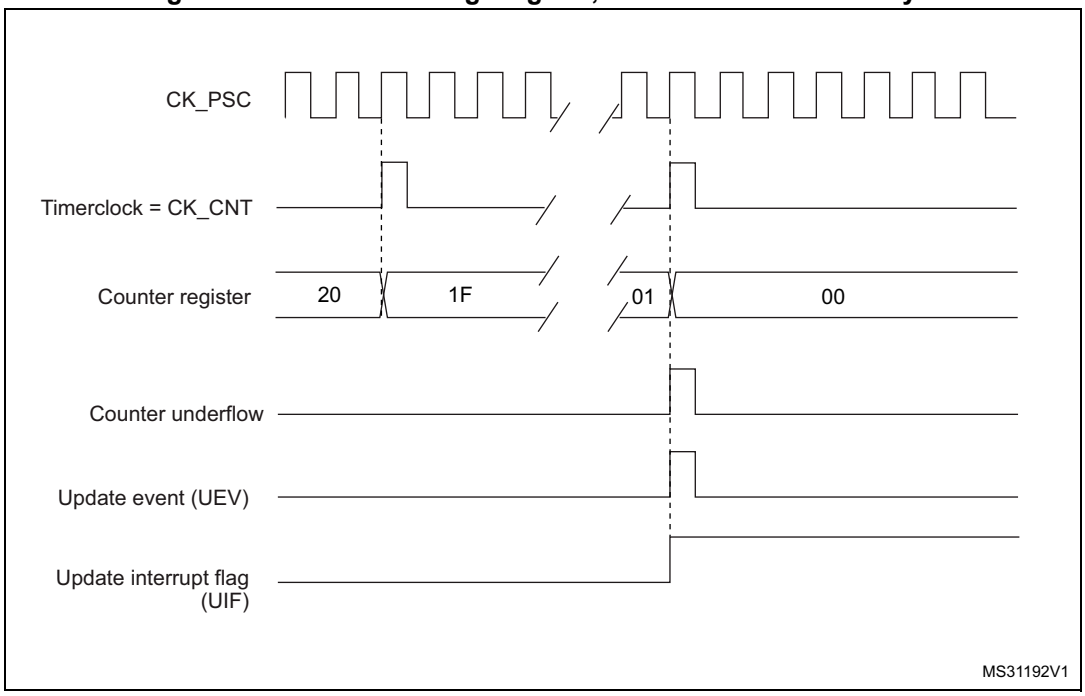


Figure 307. Counter timing diagram, Update event with ARPE=1 (counter underflow)

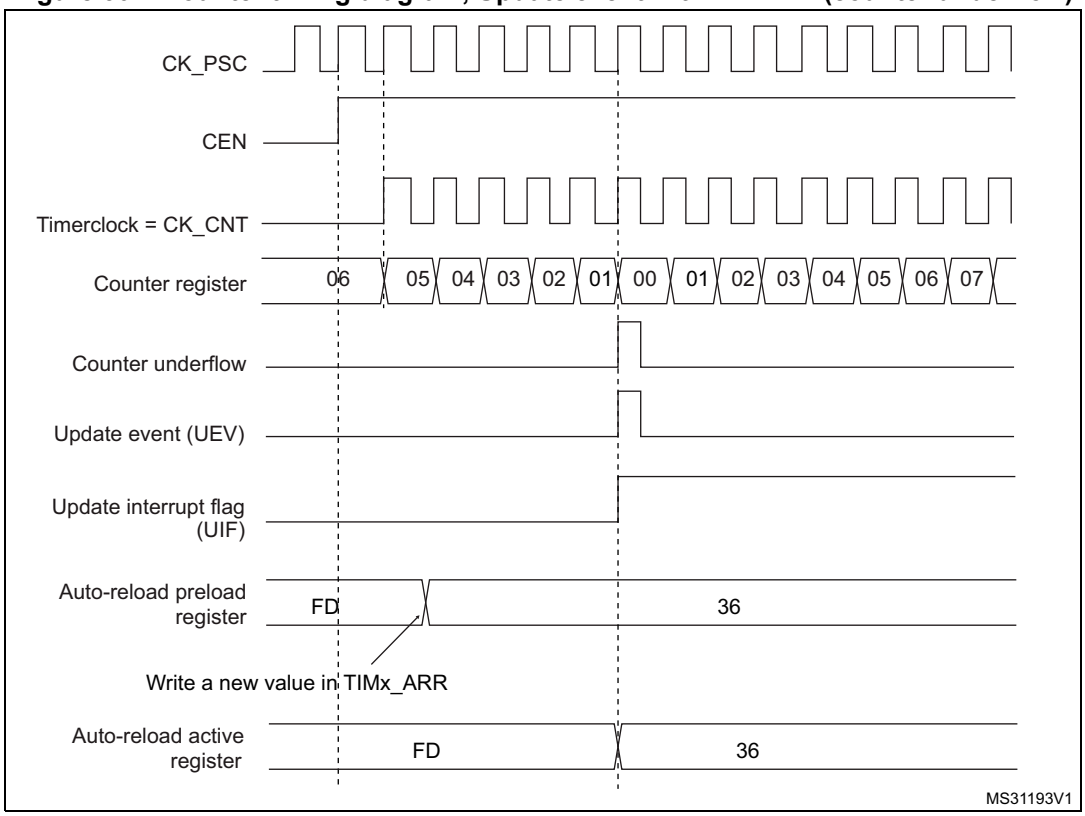
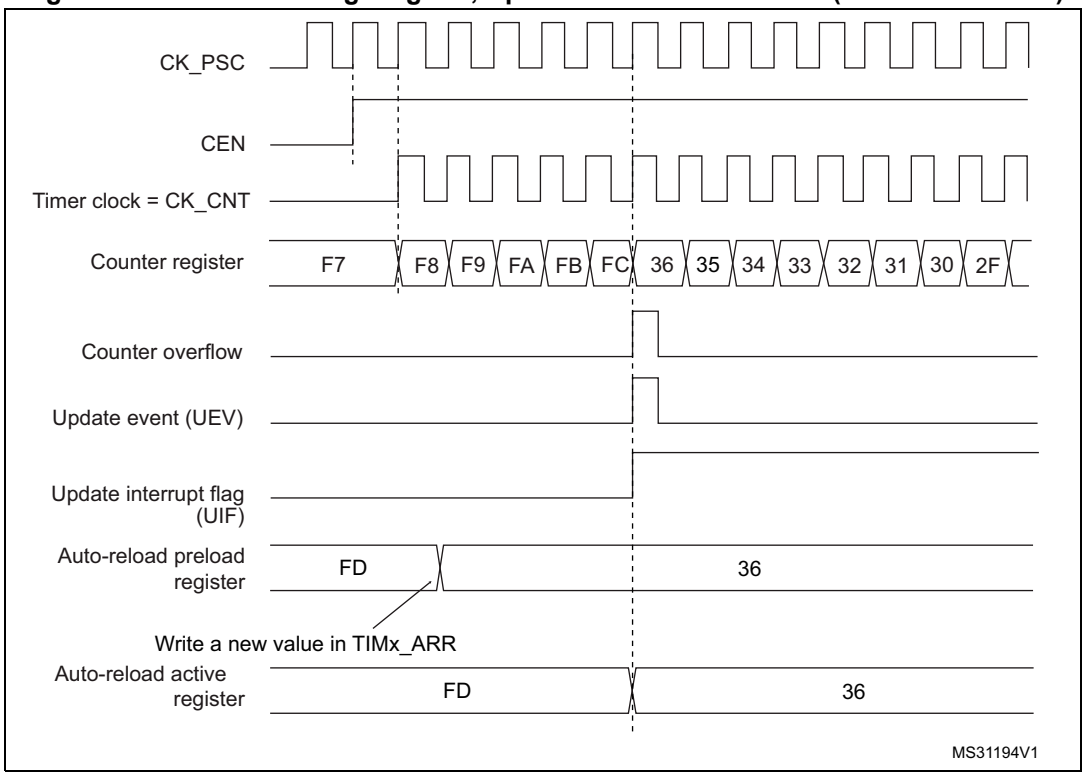


Figure 308. Counter timing diagram, Update event with ARPE=1 (counter overflow)



34.3.3 Clock selection

The counter clock can be provided by the following clock sources:

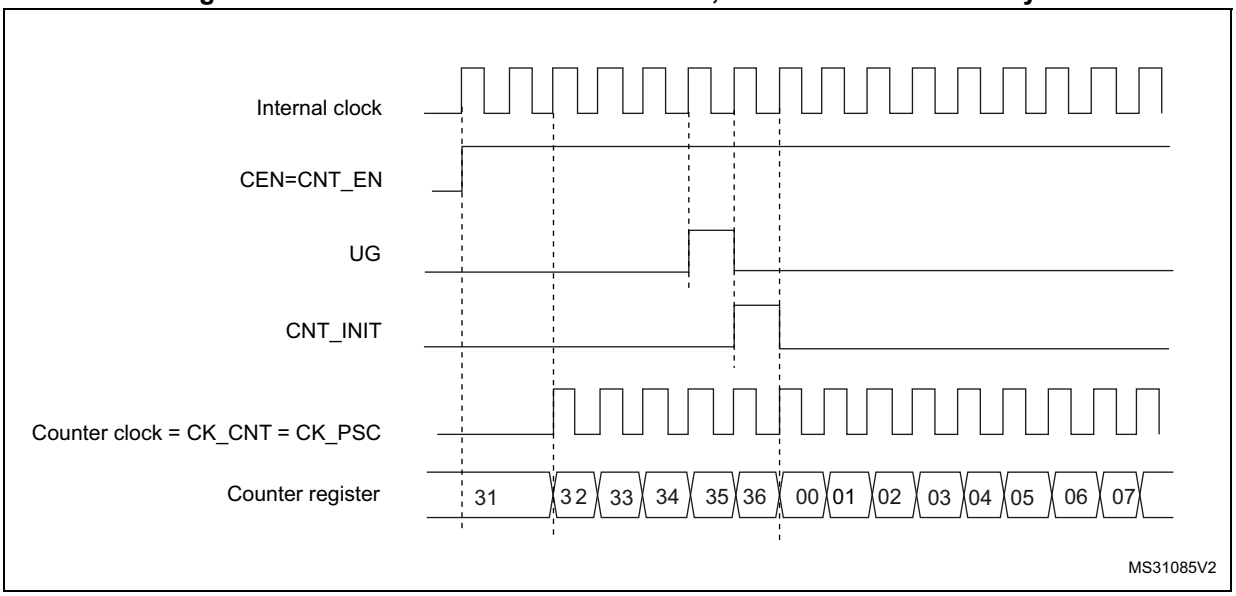
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to *: Using one timer as prescaler for another timer on page 1221* for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 309 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

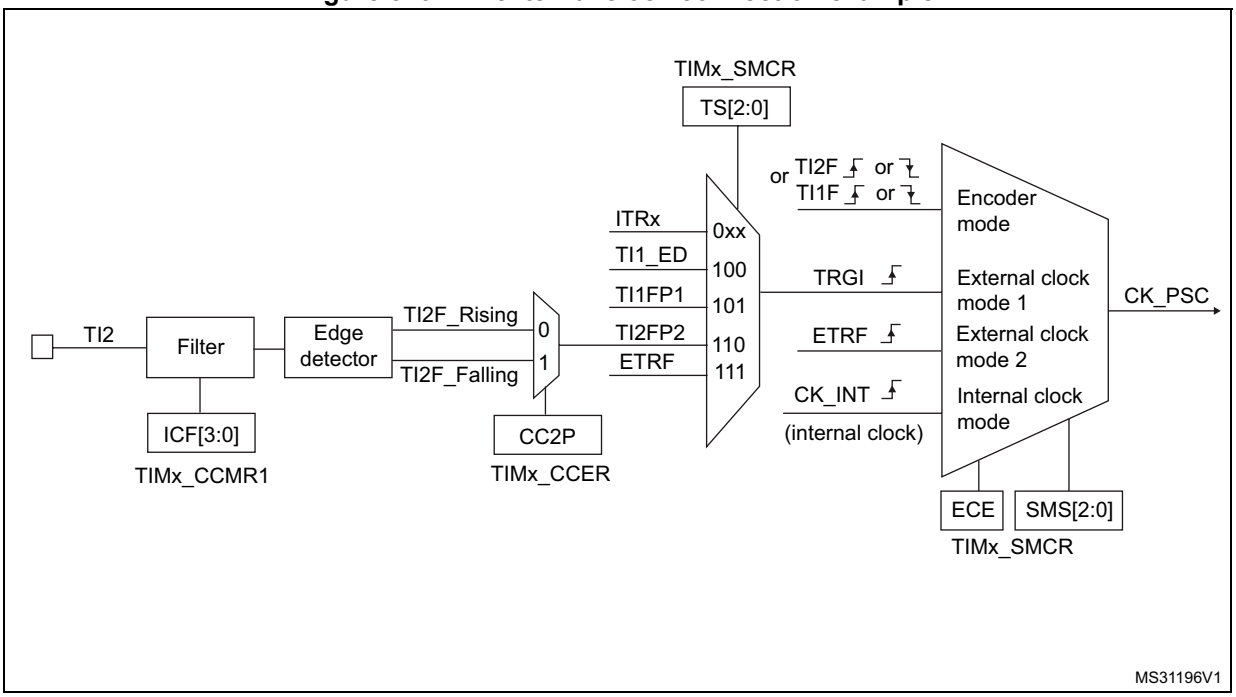
Figure 309. Control circuit in normal mode, internal clock divided by 1



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 310. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the 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).

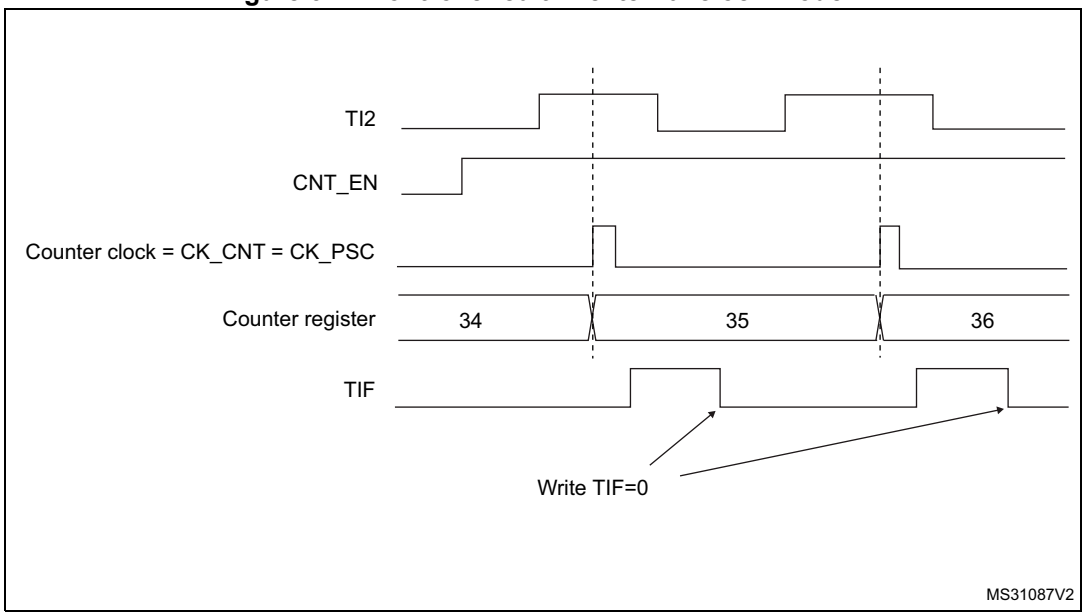
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

3. Select rising edge polarity by writing CC2P=0 and CC2NP=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 TI2 as the input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 311. Control circuit in external clock mode 1



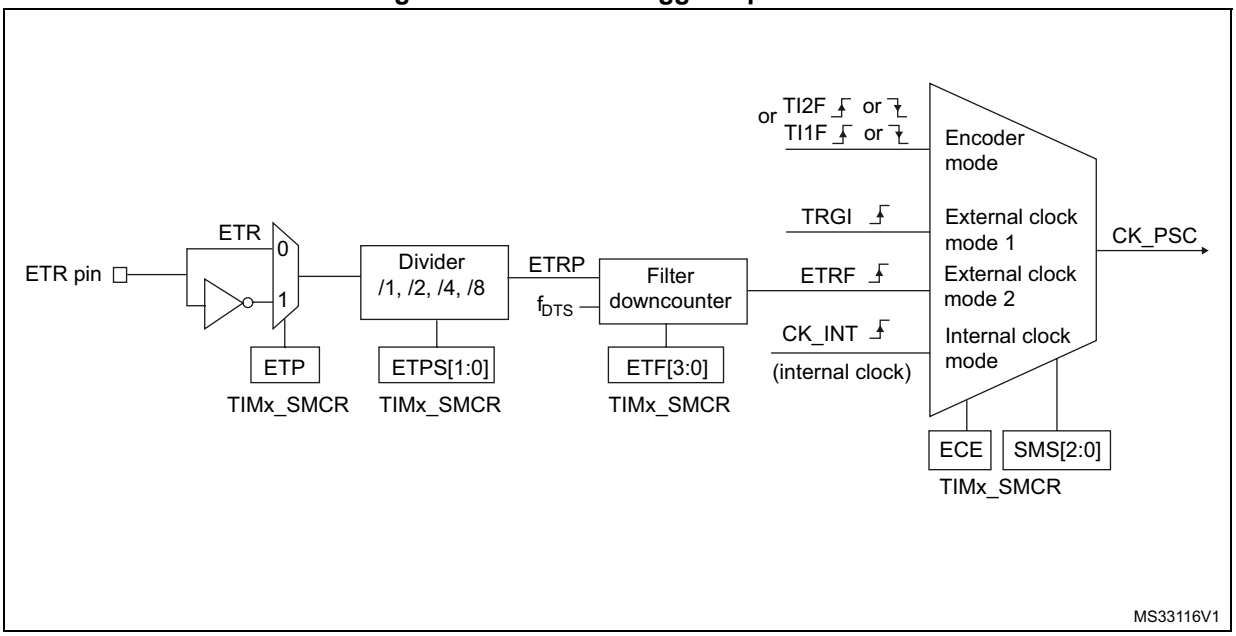
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

Figure 312 gives an overview of the external trigger input block.

Figure 312. External trigger input block



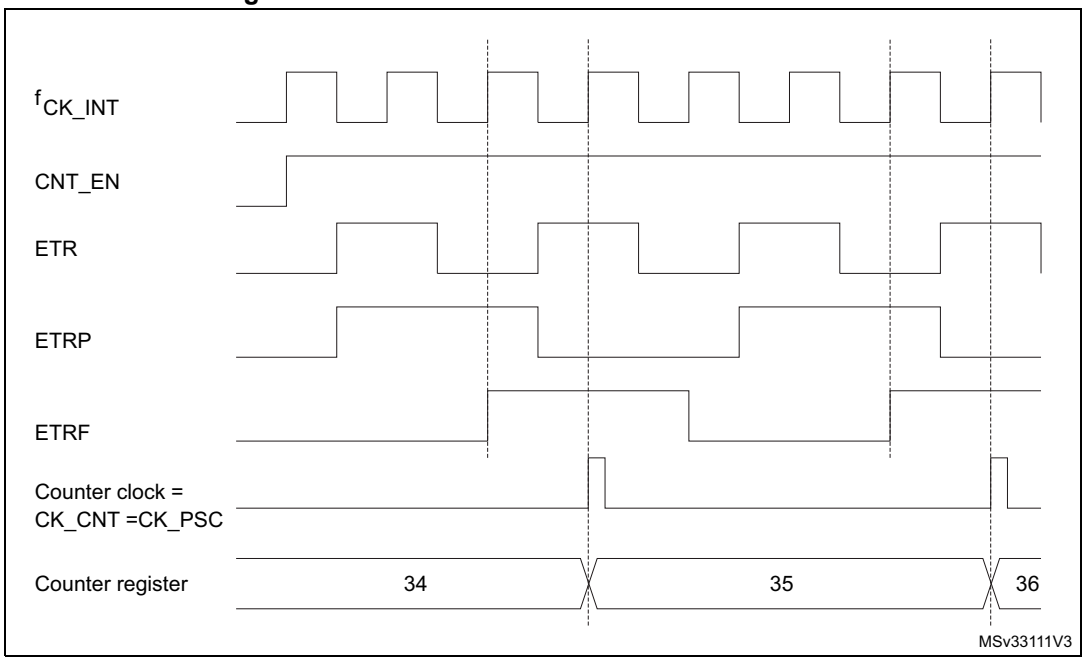
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.

Figure 313. Control circuit in external clock mode 2



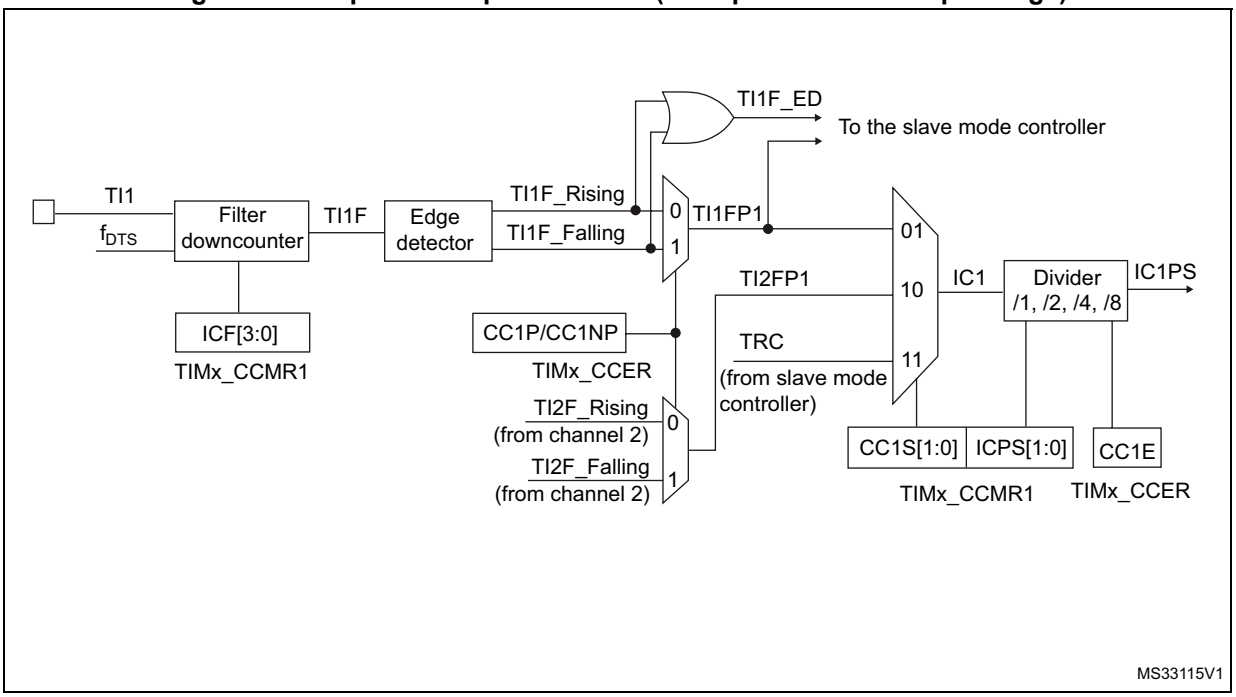
34.3.4 Capture/Compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 314. Capture/Compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 315. Capture/Compare channel 1 main circuit

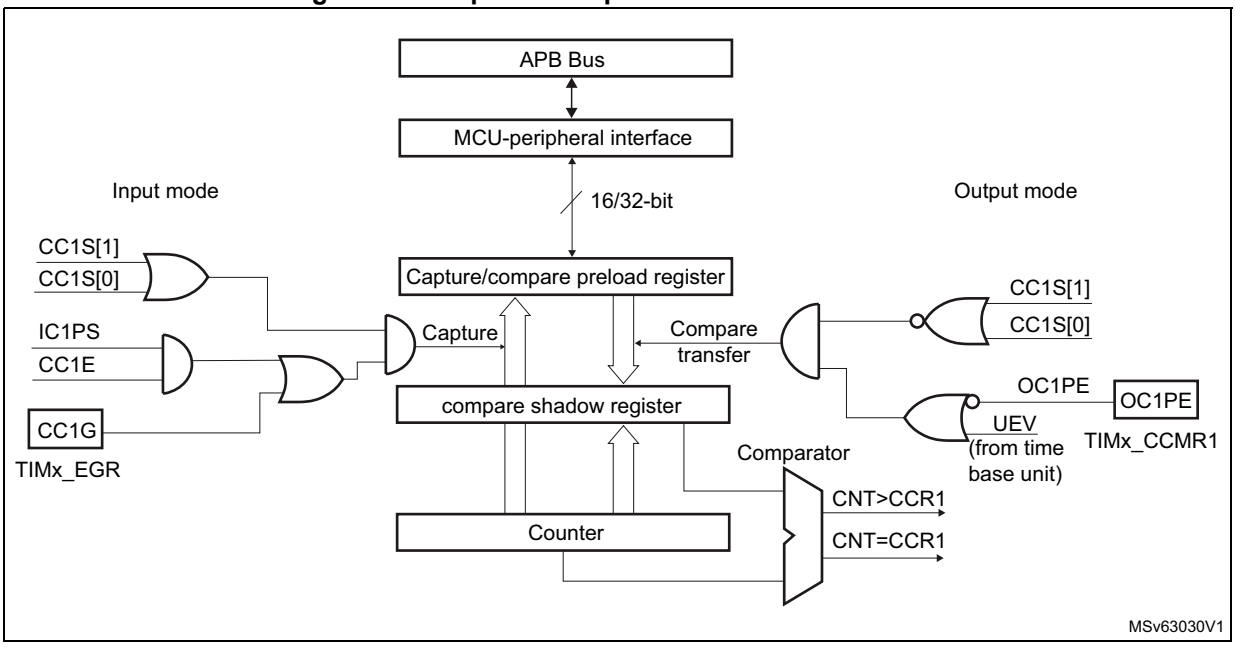
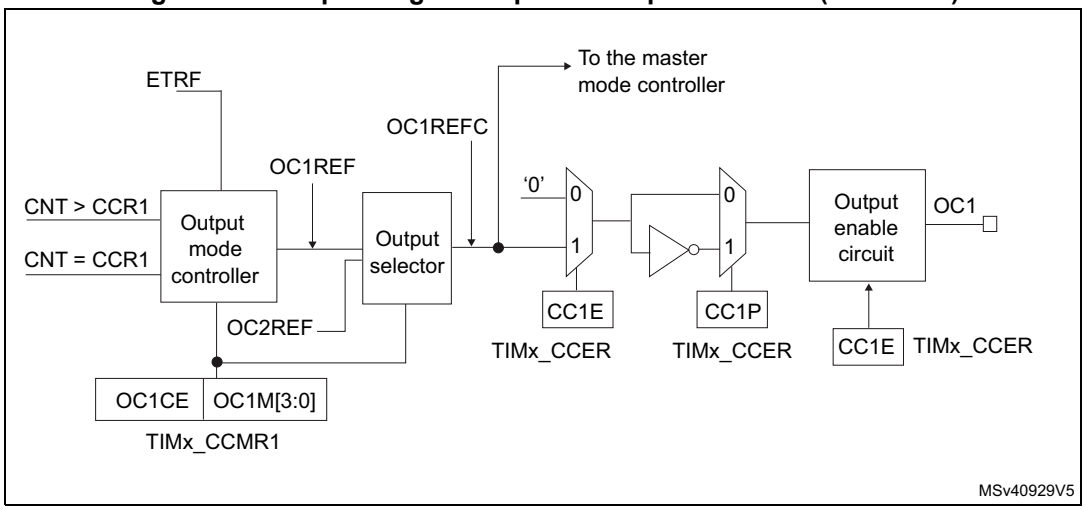


Figure 316. Output stage of Capture/Compare channel (channel 1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

34.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

3. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
4. 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).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. 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.

34.3.6 PWM input mode

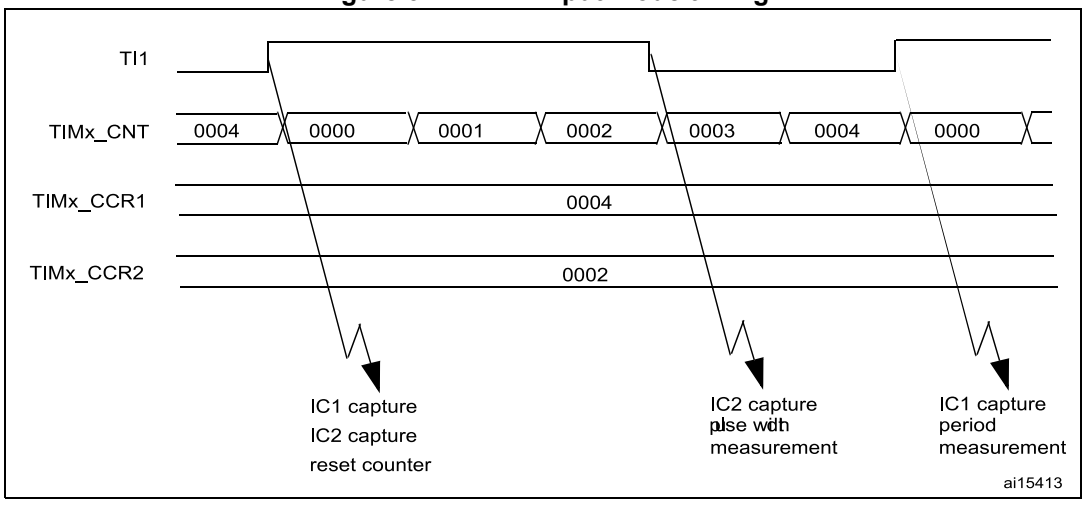
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to '0' and the CC1NP bit to '0' (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' and the CC2NP bit to '0' (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1 in the TIMx_CCER register.

Figure 317. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

34.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

34.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

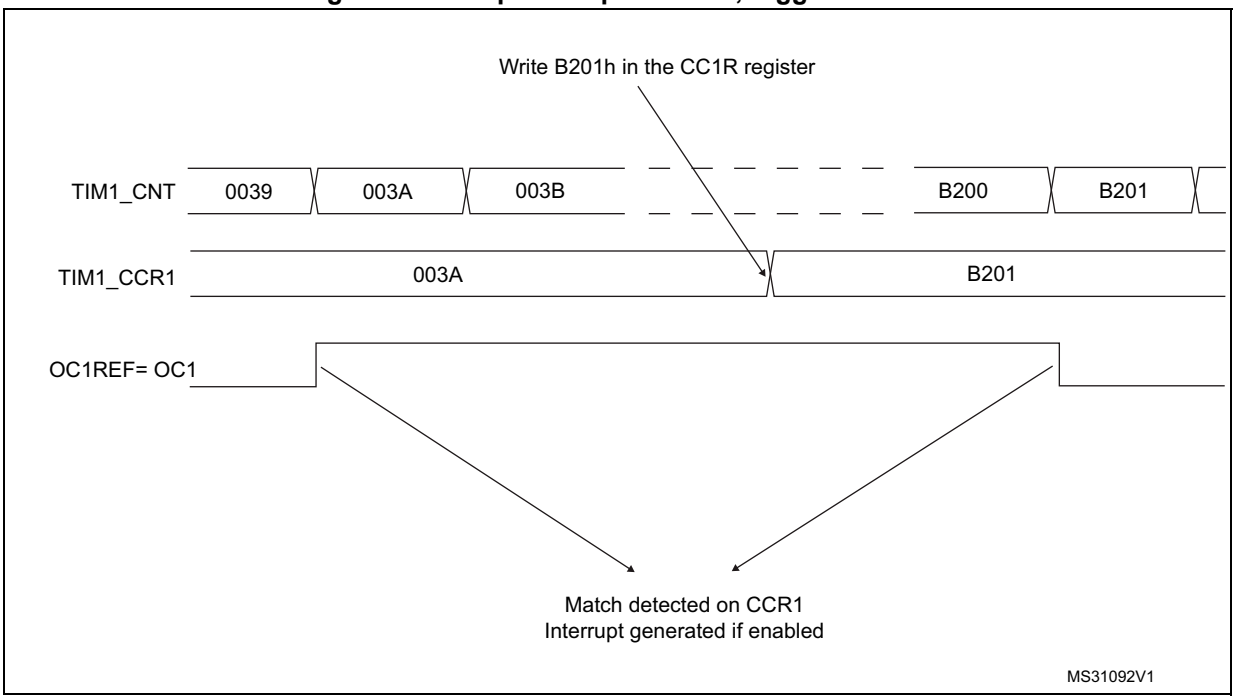
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, one must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 318*.

Figure 318. Output compare mode, toggle on OC1



34.3.9 PWM mode

Pulse width modulation mode permits to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

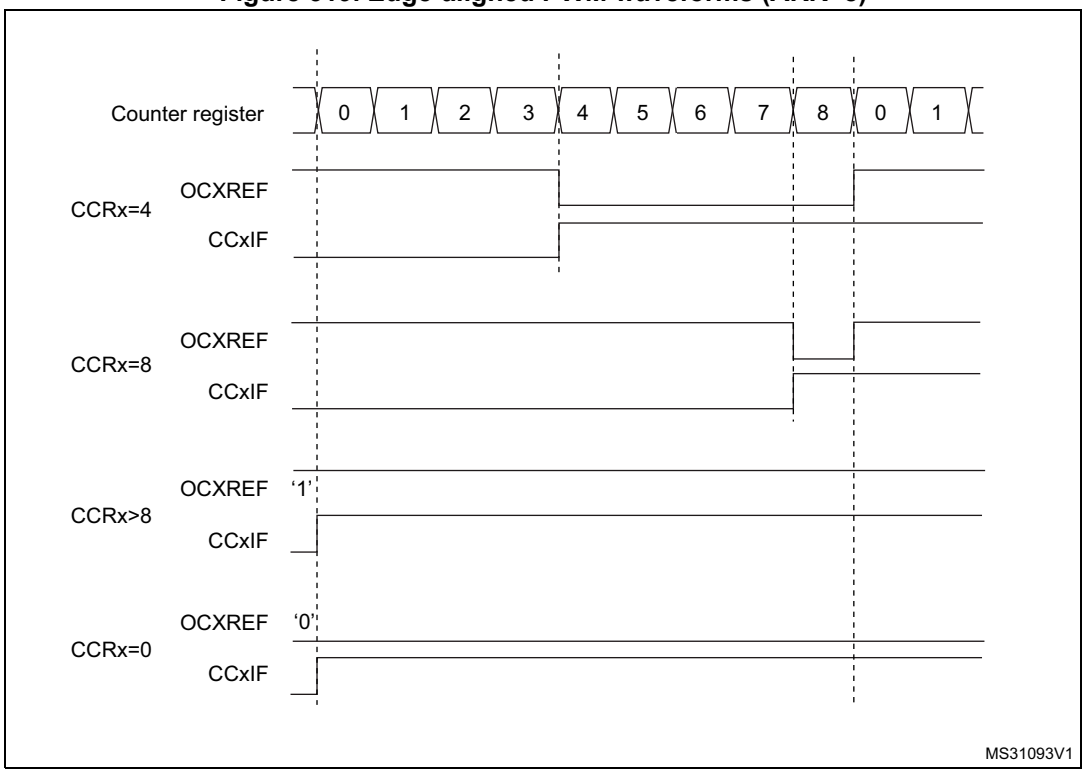
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 1185*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1. If the compare value is 0 then OCxREF is held at ‘0. *Figure 319* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 319. 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 1188*.

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

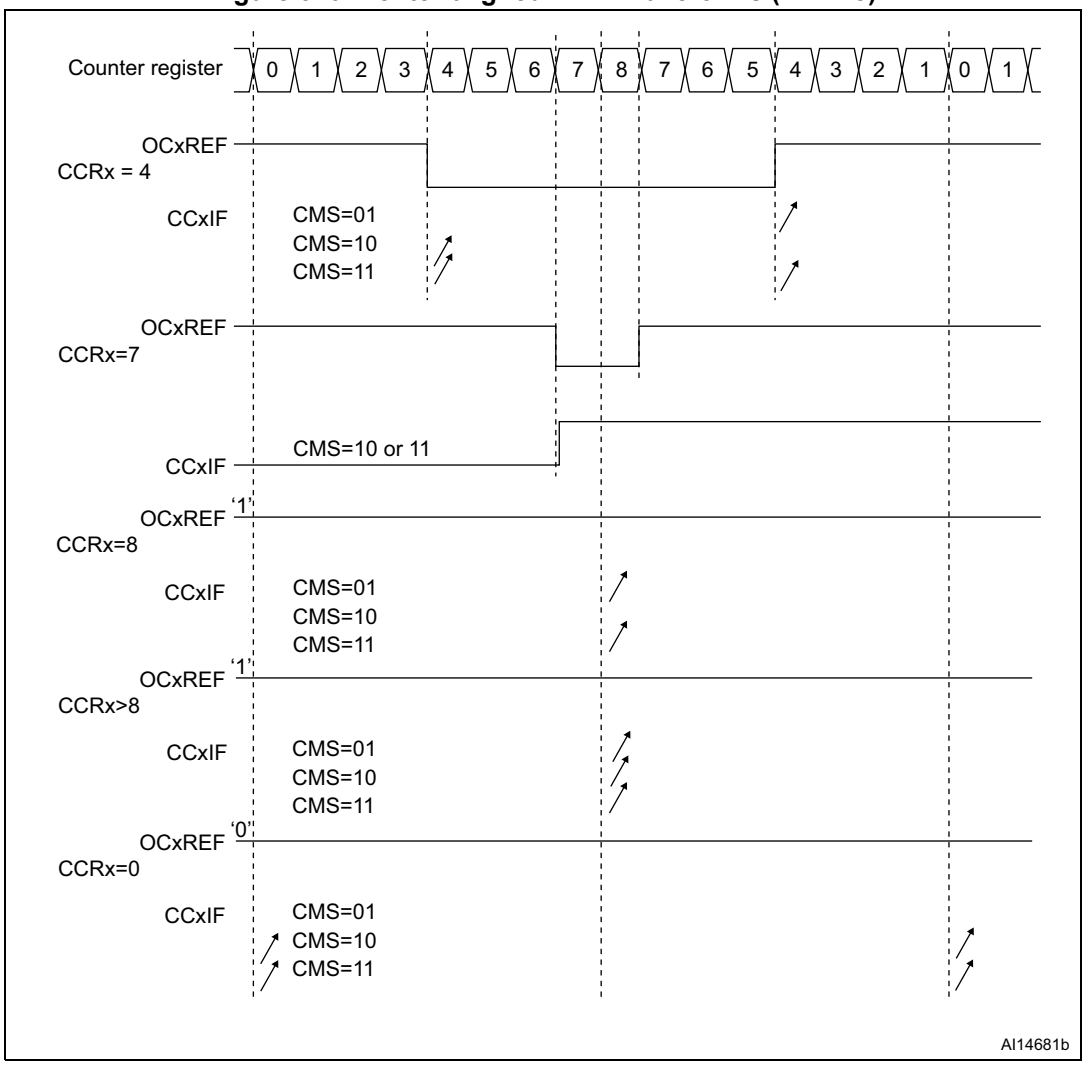
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The

compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting) on page 1191*.

Figure 320 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 320. 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.

34.3.10 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

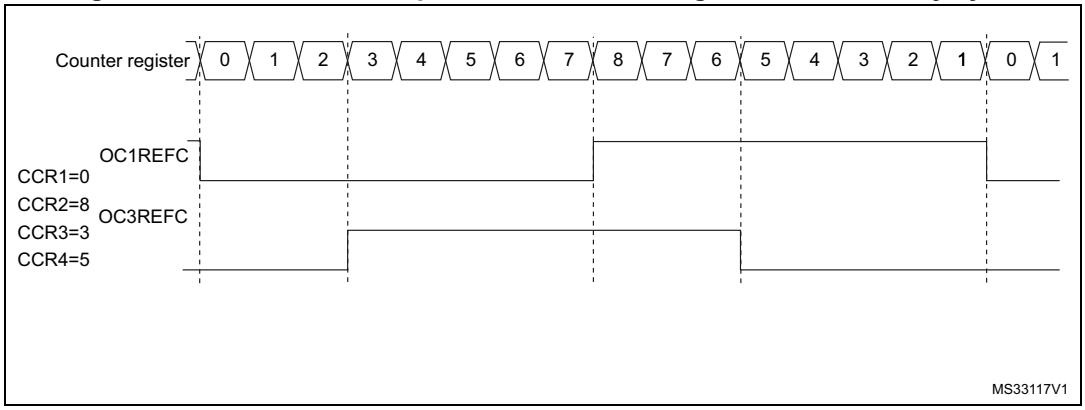
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 2.

Figure 321 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

Figure 321. Generation of 2 phase-shifted PWM signals with 50% duty cycle



34.3.11 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

– OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2

– OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

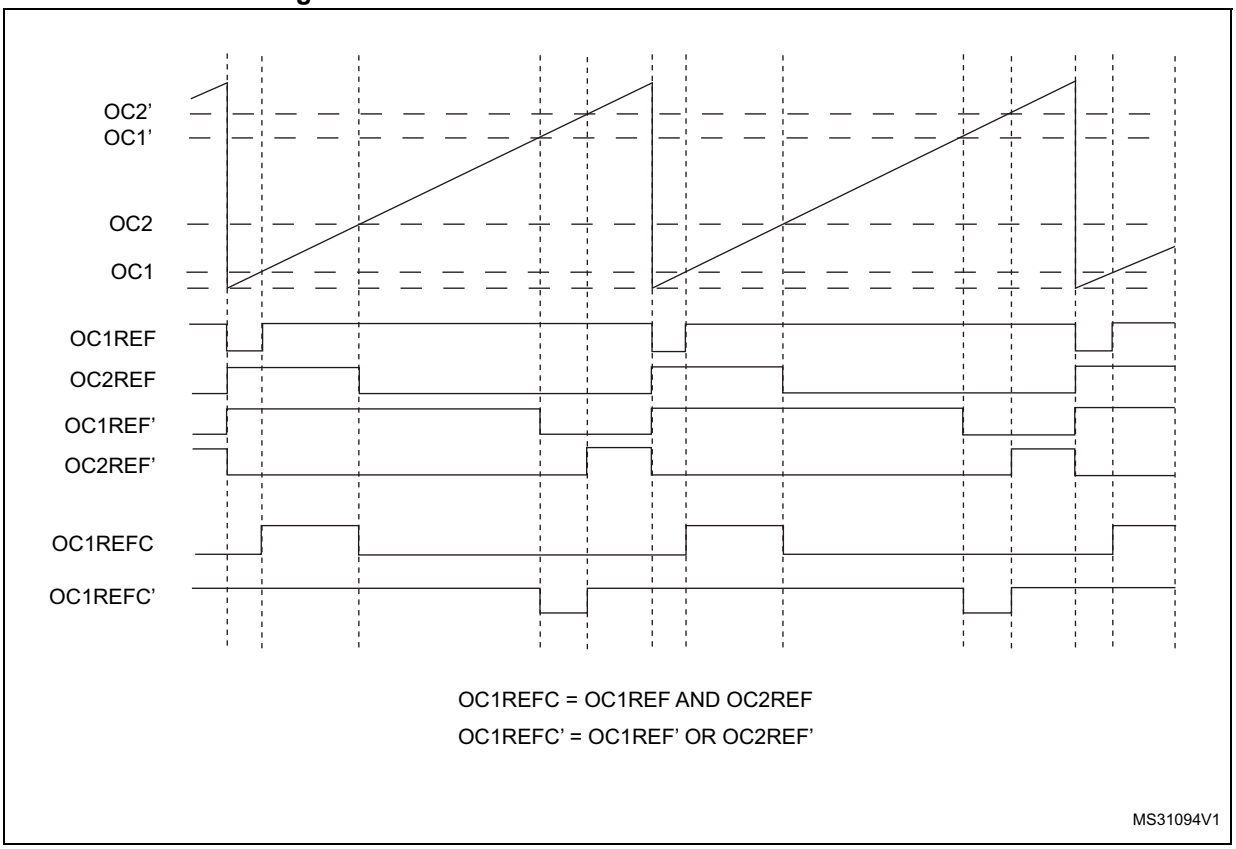
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 322 shows an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

Figure 322. Combined PWM mode on channels 1 and 3



34.3.12 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The ocref_clr_int is connected to the ETRF signal (ETR after filtering).

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next update event (UEV) occurs.

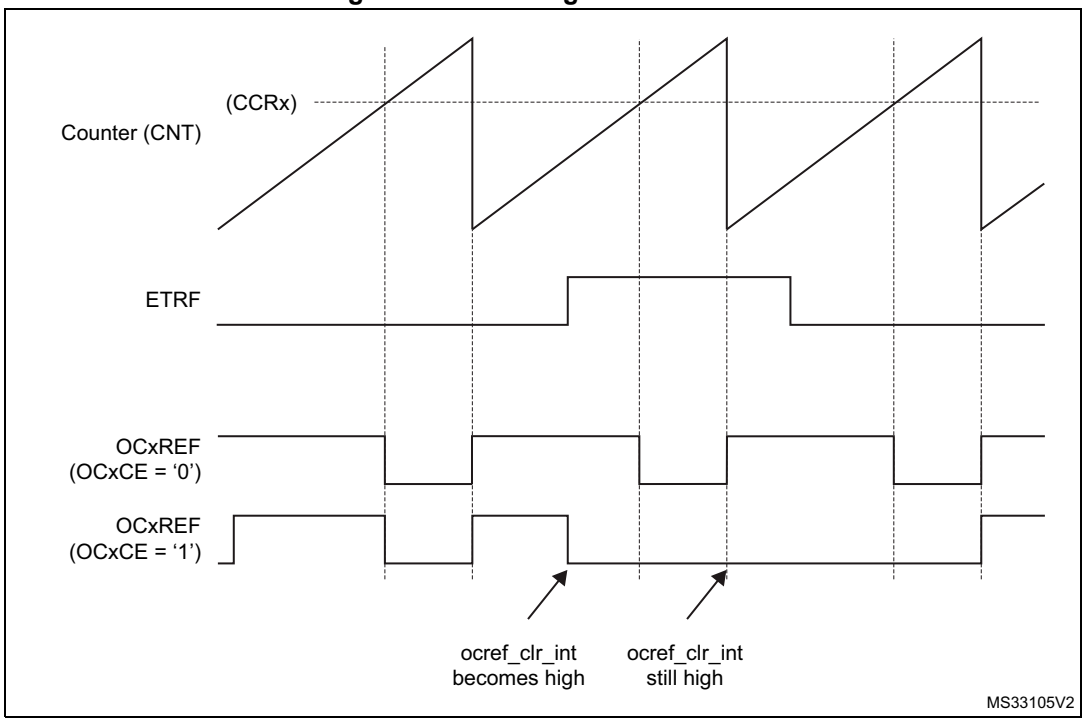
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 323 shows the behavior of the OCxREF signal when the ETRF input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 323. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

34.3.13 One-pulse mode

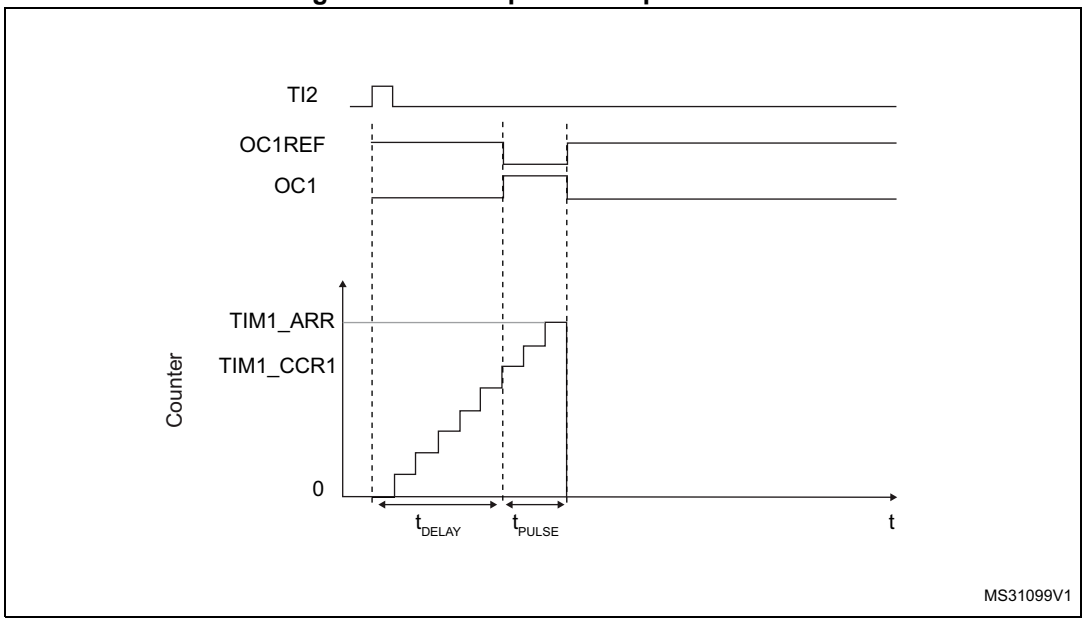
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT<CCRx ≤ ARR (in particular, 0<CCRx),

Figure 324. Example of one-pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

34.3.14 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 34.3.13*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

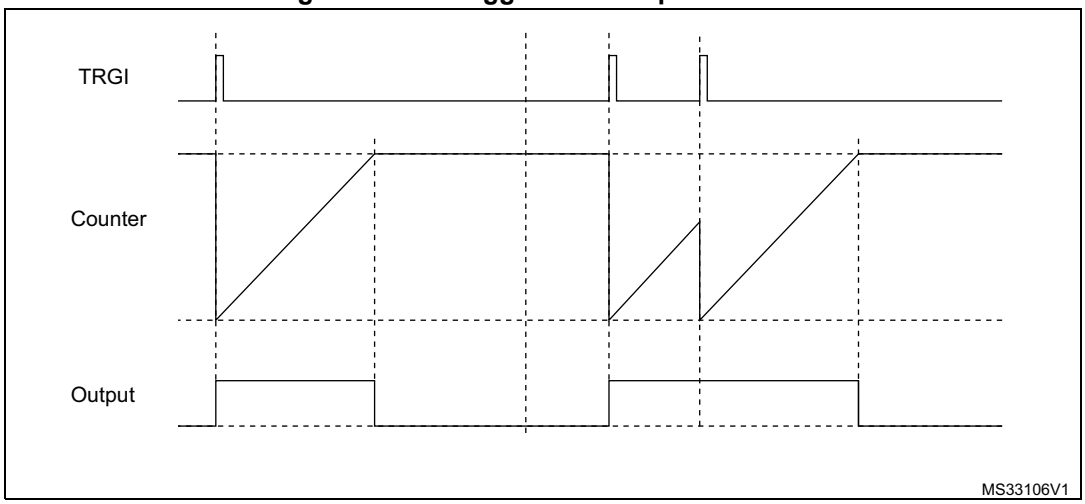
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

Note: In retriggerable one pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 325. Retriggerable one-pulse mode.



34.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 274*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 274. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Figure 326 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = '0' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = '0' (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

Figure 326. Example of counter operation in encoder interface mode

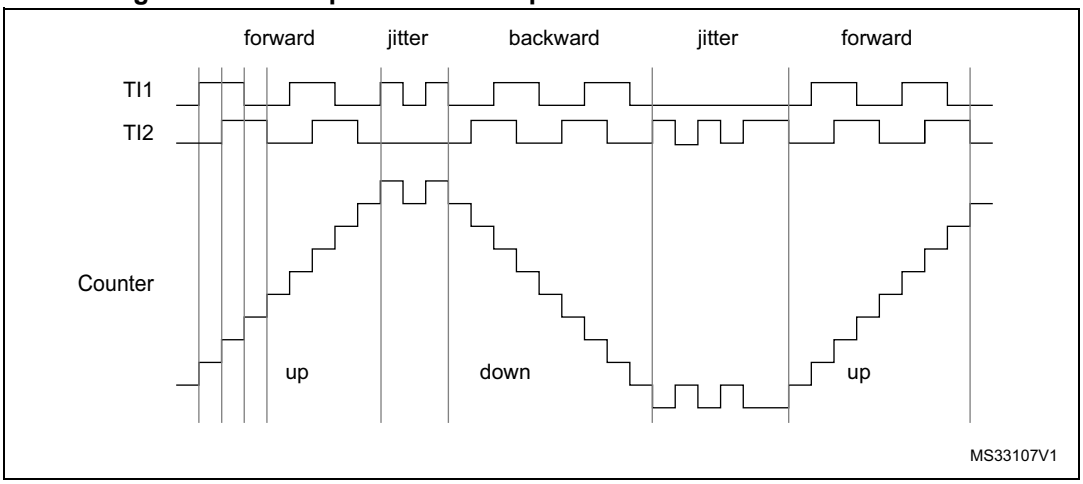
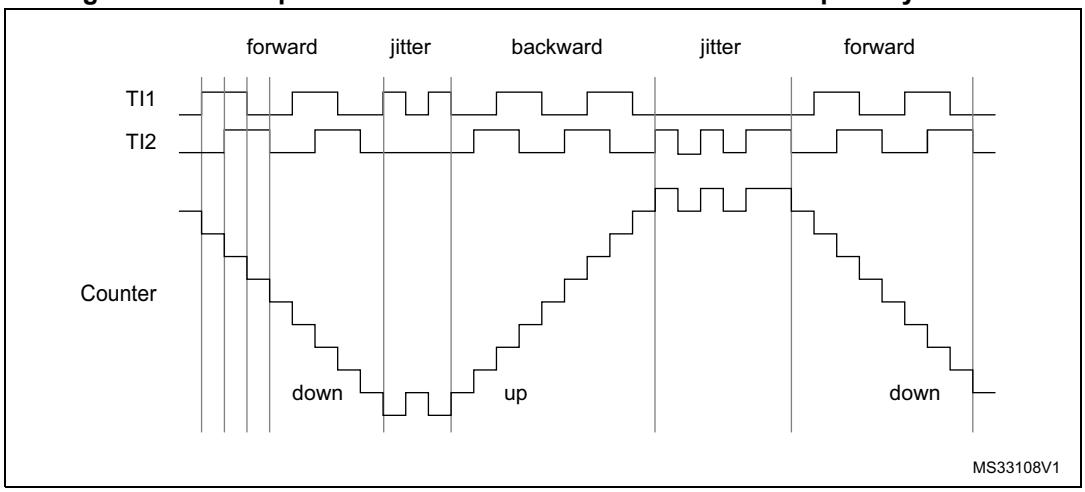


Figure 327 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 327. Example of encoder interface mode with TI1FP1 polarity inverted



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

34.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This permits to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

34.3.17 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 33.3.25: Interfacing with Hall sensors on page 1127*.

34.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

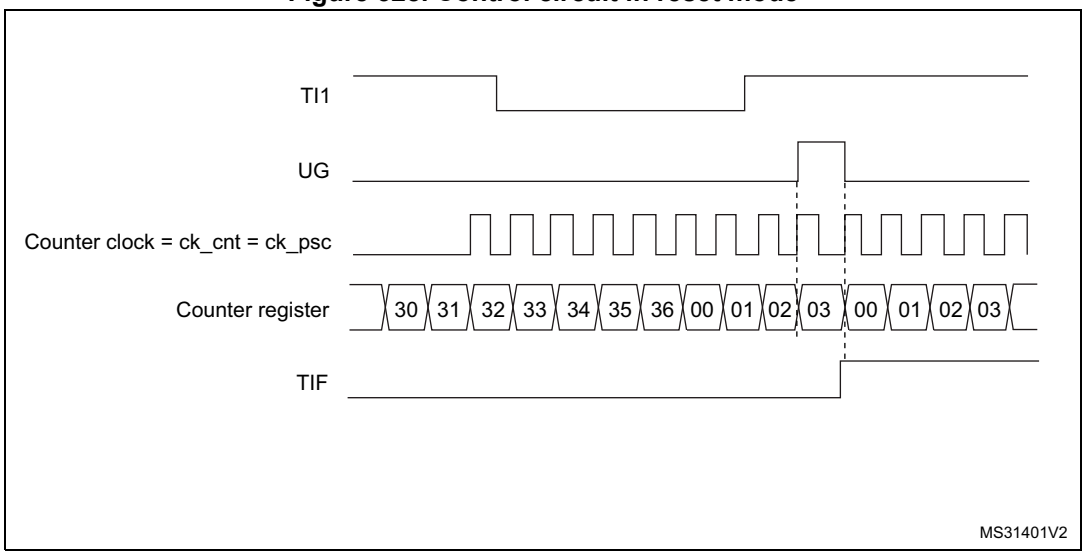
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 328. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

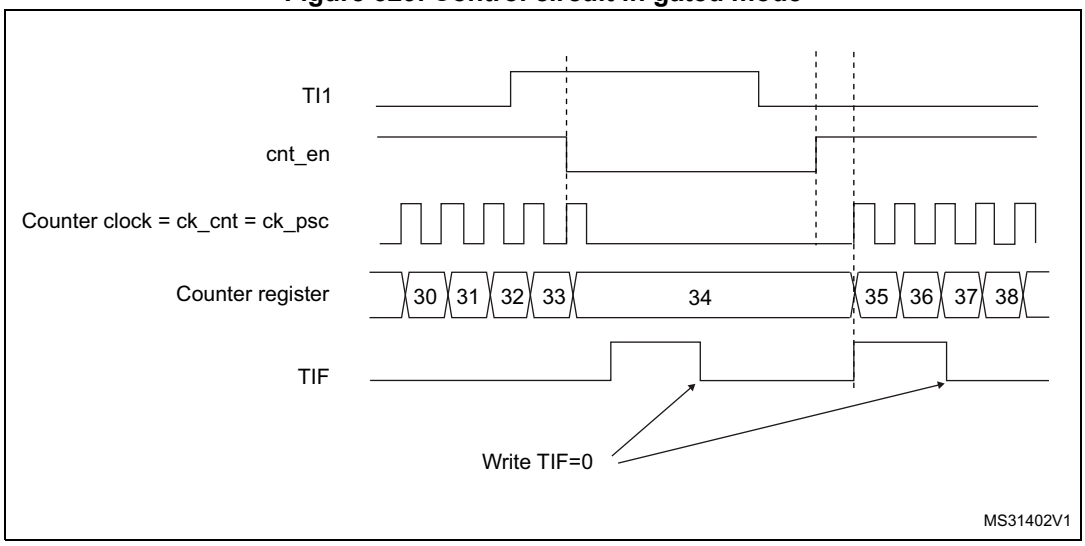
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 329. Control circuit in gated mode



1. The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Note: The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write

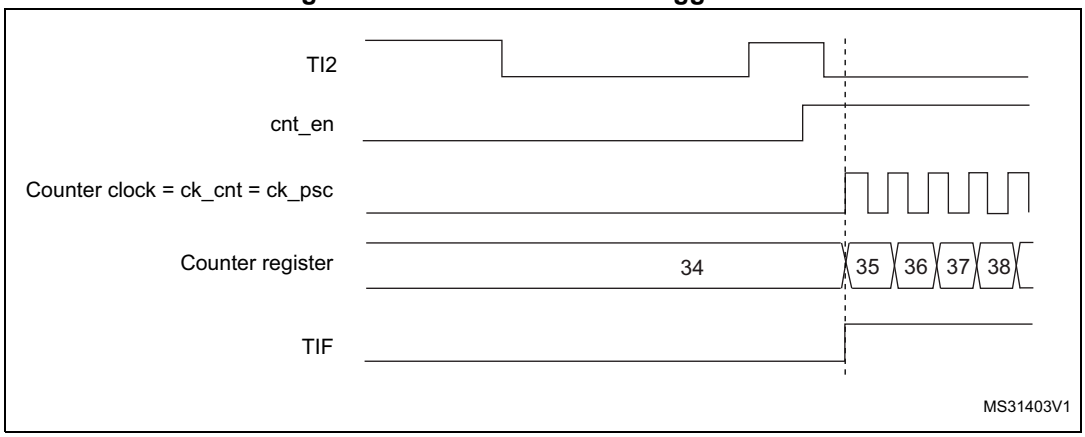
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 330. Control circuit in trigger mode



Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

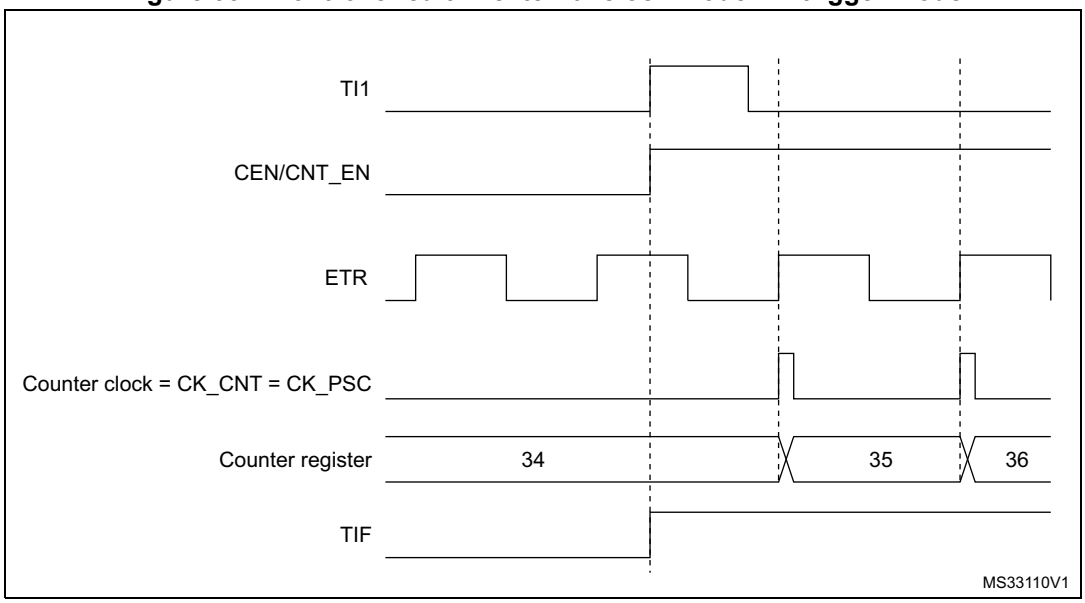
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 331. Control circuit in external clock mode 2 + trigger mode



34.3.19 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 332: Master/Slave timer example and *Figure 333: Master/slave connection example with 1 channel only timers* present an overview of the trigger selection and the master mode selection blocks.

Figure 332. Master/Slave timer example

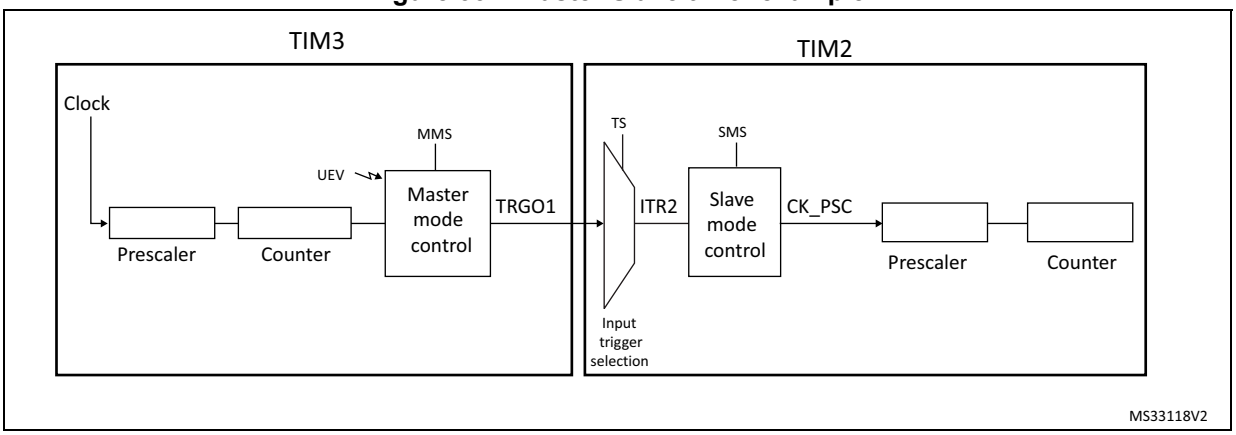
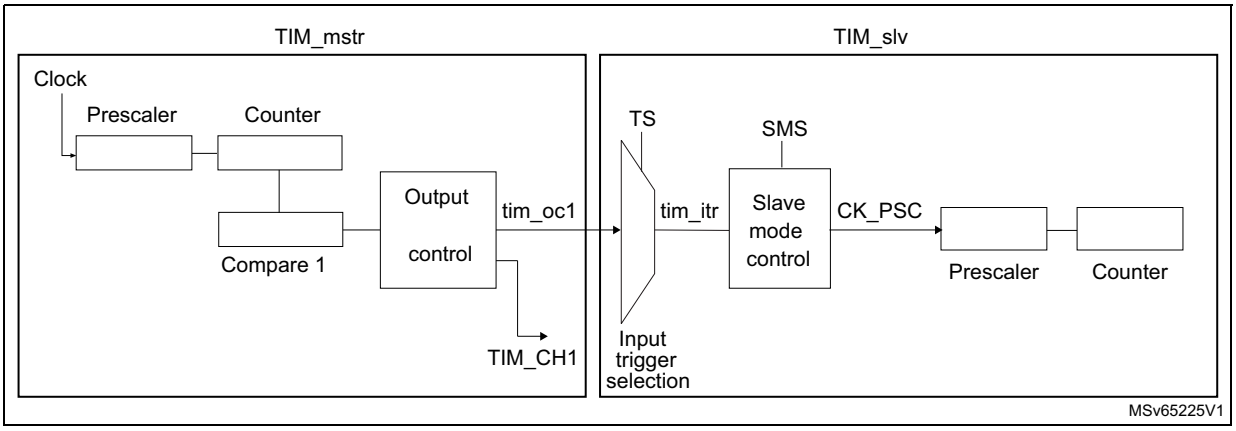


Figure 333. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 333) do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the "TIMx internal trigger connection" table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.
The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger.
For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

Using one timer as prescaler for another timer

For example, TIM3 can be configured to act as a prescaler for TIM2. Refer to *Figure 332*. To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM3_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM2_SMCR register (writing TS=010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

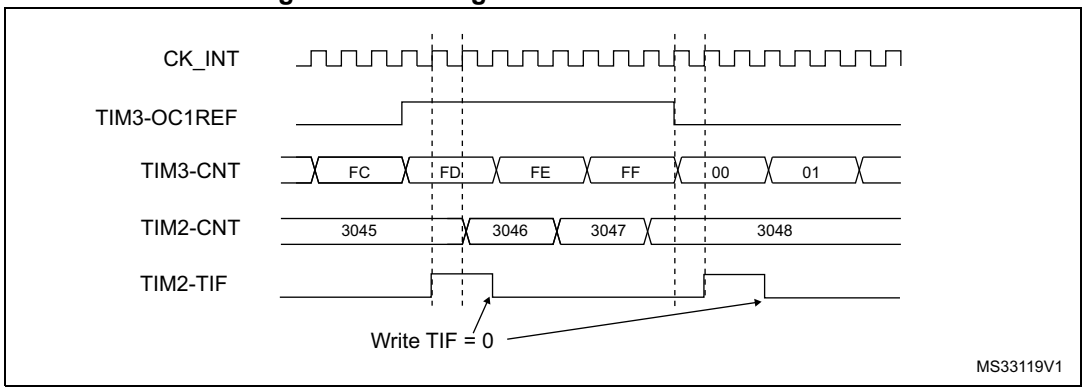
Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to *Figure 332* for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
6. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

Figure 334. Gating TIM2 with OC1REF of TIM3

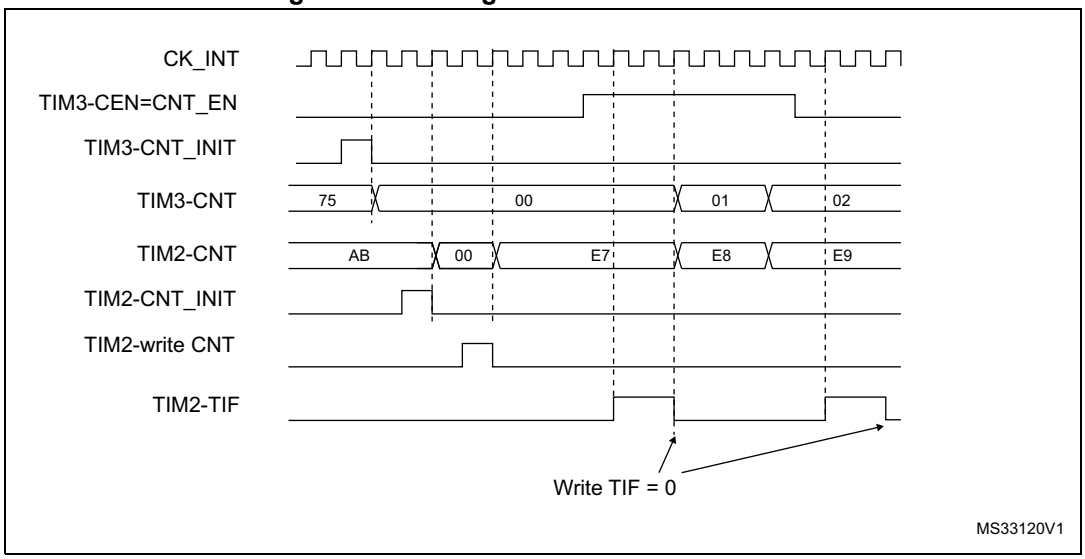


In the example in *Figure 334*, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 335*), we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing ‘0 to the CEN bit in the TIM3_CR1 register:

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM3 by writing ‘1 in UG bit (TIM3_EGR register).
6. Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
7. Initialize TIM2 to 0xE7 by writing ‘0xE7’ in the TIM2 counter (TIM2_CNTL).
8. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
9. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).
10. Stop TIM3 by writing ‘0 in the CEN bit (TIM3_CR1 register).

Figure 335. Gating TIM2 with Enable of TIM3

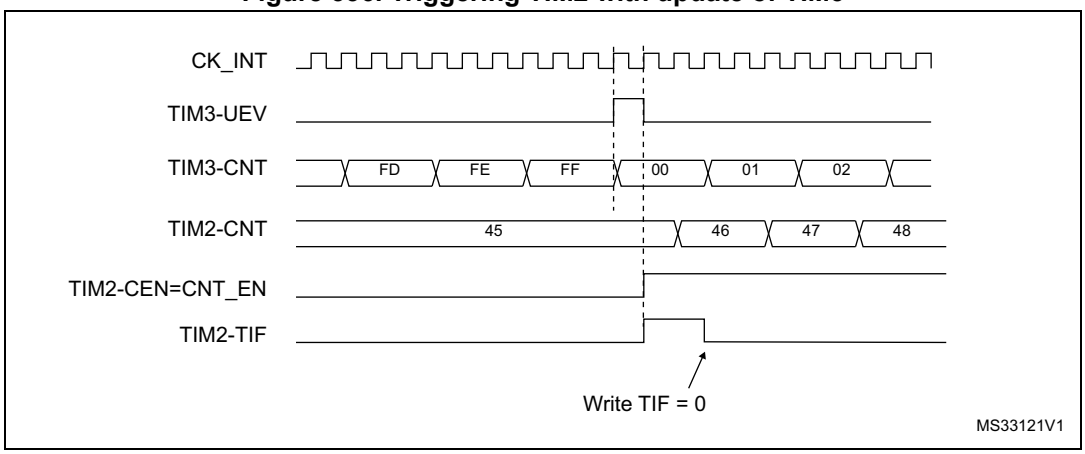


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to *Figure 332* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

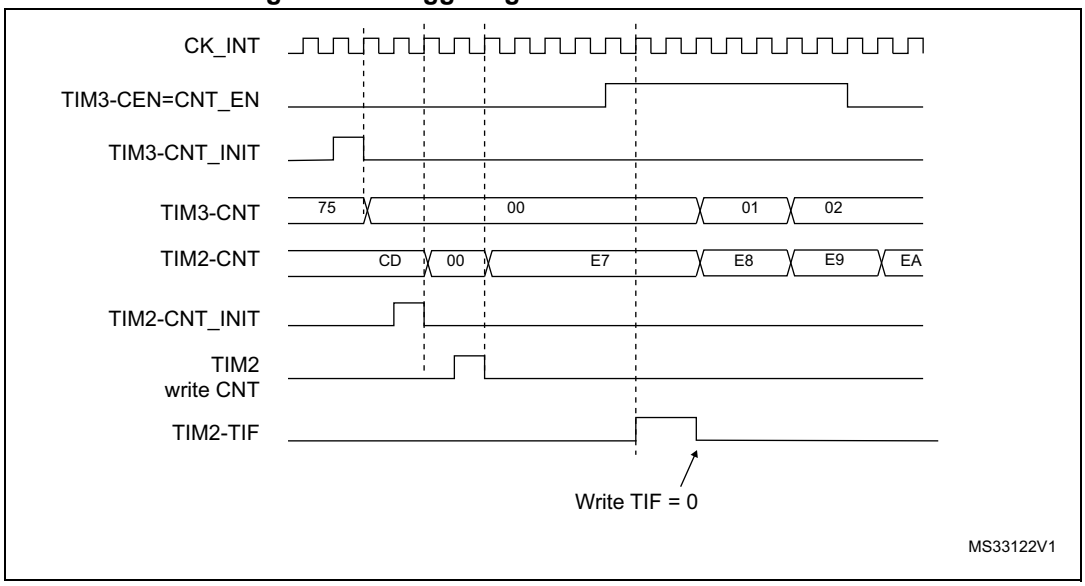
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
2. Configure the TIM3 period (TIM3_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=010 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Figure 336. Triggering TIM2 with update of TIM3



As in the previous example, both counters can be initialized before starting counting. *Figure 337* shows the behavior with the same configuration as in *Figure 336* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 337. Triggering TIM2 with Enable of TIM3



Starting 2 timers synchronously in response to an external trigger

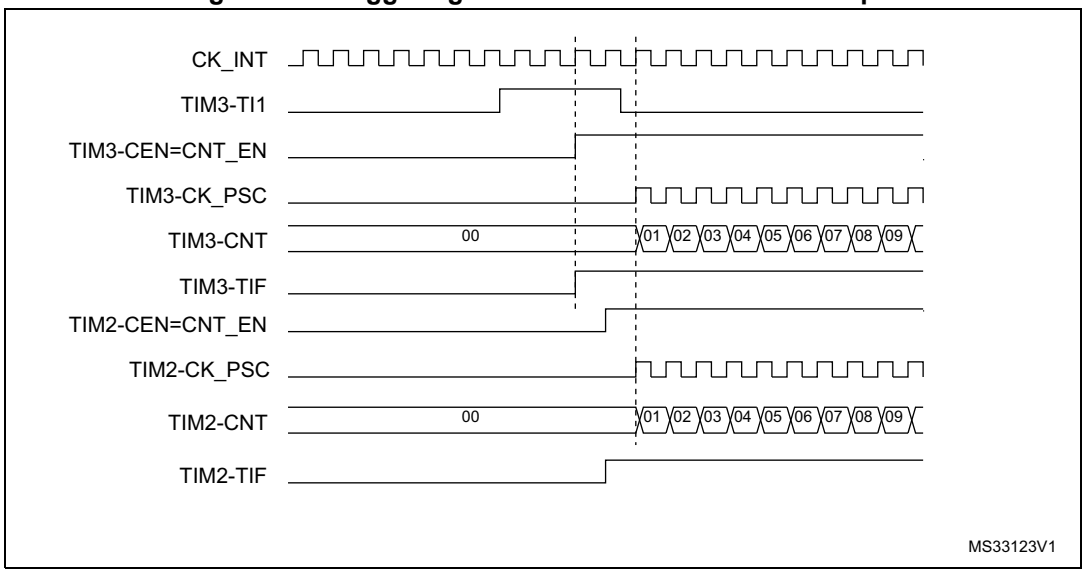
In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to *Figure 332* for connections. To ensure the counters are aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

1. Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3_CR2 register).
2. Configure TIM3 slave mode to get the input trigger from TI1 (TS=100 in the TIM3_SMCR register).
3. Configure TIM3 in trigger mode (SMS=110 in the TIM3_SMCR register).
4. Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3_SMCR register).
5. Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
6. Configure TIM2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (TIM3), 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 TIM3.

Figure 338. Triggering TIM3 and TIM2 with TIM3 TI1 input



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

34.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

34.3.21 Debug mode

When the microcontroller enters debug mode (Cortex®-M33 core - halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module. For more details, refer to *Section : DBGMCU APB1 freeze register 1 (DBGMCU_APB1FZR1)*.

34.4 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).

34.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

34.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

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

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
See also *Section 33.3.25: Interfacing with Hall sensors on page 1127*

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits permit to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)
100: **Compare** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

34.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

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.	SMS[3]
															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]			Res.	SMS[2:0]		
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 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.
000: Internal Trigger 0 (ITR0).
001: Internal Trigger 1 (ITR1).
010: Internal Trigger 2 (ITR2).
011: Internal Trigger 3 (ITR3).
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)
111: External Trigger input (ETRF)
See *Table 275: TIMx internal trigger connection on page 1232* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 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 (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.
0000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.
0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 275. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	TIM8/USB_SOF(1)	TIM3	TIM4
TIM3	TIM1	TIM2	TIM15	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

1. Depends on the bit ITR1_RMP in TIM2_OR1 register.

34.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x0C

Reset value: 0x0000

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

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 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.

34.4.5 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow (for TIM2 to TIM4) 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.

34.4.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

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

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

34.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

34.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

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, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

34.4.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 5)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

34.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 5)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

34.4.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*
CC1 channel configured as output: CC1NP must be kept cleared in this case.
CC1 channel configured as input: This bit is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*
0: Capture mode disabled / OC1 is not active
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 276. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO and AFIO registers.

34.4.12 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 5)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- This section is for UIFREMAP = 0
- Next section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 31:16 **CNT[31:16]**: Most significant part counter value (TIM2 and TIM5)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

34.4.13 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 5)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- Previous section is for UIFREMAP = 0
- This section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:16 **CNT[30:16]**: Most significant part counter value (TIM2 and TIM5)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

34.4.14 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

34.4.15 TIMx auto-reload register (TIMx_ARR)(x = 2 to 5)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

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

Bits 31:16 **ARR[31:16]**: High auto-reload value (TIM2 and TIM5)

Bits 15:0 **ARR[15:0]**: Low Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 34.3.1: Time-base unit on page 1183* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

34.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR1[31:16]**: High Capture/Compare 1 value (TIM2 and TIM5)

Bits 15:0 **CCR1[15:0]**: Low Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

34.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (TIM2 and TIM5)

Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

34.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (TIM2 and TIM5)

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

If channel CC3 is configured as output:
CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.
If channel CC3is configured as input:
CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

34.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (TIM2 and TIM5)

Bits 15:0 **CCR4[15:0]**: Low Capture/Compare value

1. if CC4 channel is configured as output (CC4S bits):
   CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).
   It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
   The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.
2. if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):
   CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

34.4.20 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...
Example: Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

34.4.21 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

34.4.22 TIM2 option register 1 (TIM2_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI4_RMP[1:0]		ETR_ RMP	ITR1_ RMP
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **TI4_RMP[1:0]**: Input Capture 4 remap
00: TIM2 input capture 4 is connected to I/O
01: TIM2 input capture 4 is connected to COMP1_OUT
10: TIM2 input capture 4 is connected to COMP2_OUT
11: TIM2 input capture 4 is connected to logical OR between COMP1_OUT and COMP2_OUT

Bit 1 **ETR_RMP**: External trigger remap
0: TIM2_ETR is connected to I/O
1: TIM2_ETR is connected to LSE

Bit 0 **ITR1_RMP**: Internal trigger 1 remap
0: TIM2_ITR1 is connected to TIM8_TRGO
1: TIM2_ITR1 is connected to USB_SOF

34.4.23 TIM3 option register 1 (TIM3_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TI1_RMP[1:0]**: Input Capture 1 remap
00: TIM3 input capture 1 is connected to I/O
01: TIM3 input capture 1 is connected to COMP1_OUT
10: TIM3 input capture 1 is connected to COMP2_OUT
11: TIM3 input capture 1 is connected to logical OR between COMP1_OUT and COMP2_OUT

34.4.24 TIM2 option register 2 (TIM2_OR2)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETR SEL2
															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 31:17 Reserved, must be kept at reset value.

Bits 16:14 **ETRSEL[2:0]**: ETR source selection
These bits select the ETR input source.
000: TIM2_ETR source is selected with the ETR_RMP bitfield in TIM2_OR1 register
001: COMP1 output connected to ETR input
010: COMP2 output connected to ETR input
Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

34.4.25 TIM3 option register 2 (TIM3_OR2)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETR SEL2
															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 31:17 Reserved, must be kept at reset value.

Bits 16:14 **ETRSEL[2:0]**: ETR source selection
These bits select the ETR input source.
000: ETR input is connected to I/O
001: COMP1 output connected to ETR input
Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

34.4.26 TIMx register map

TIMx registers are mapped as described in the table below:

Table 277. 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
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0		0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0

Table 277. 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
0x24	**TIMx_CNT**	CNT[31] or UIFCPY	CNT[30:16] (TIM2 and TIM5 only, reserved on the other timers)															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
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	ARR[31:16] (TIM2 and TIM5 only, reserved on the other timers)																ARR[15:0]															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	Reserved																																
0x34	**TIMx_CCR1**	CCR1[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR1[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
0x38	**TIMx_CCR2**	CCR2[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR2[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
0x3C	**TIMx_CCR3**	CCR3[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR3[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
0x40	**TIMx_CCR4**	CCR4[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR4[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
0x44	Reserved																																
0x48	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM2_OR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI4_RMP[1:0]		ETR_RMP	ITR1_RMP
	Reset value																													0	0	0	0

Table 277. 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
0x50	**TIM3_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.	TI1_RMP[1:0]
	Reset value																																0
0x60	**TIM2_OR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0														
0x60	**TIM3_OR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0														

Refer to *Section 2.3 on page 87* for the register boundary addresses.

35 General-purpose timers (TIM15/TIM16/TIM17)

35.1 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 35.4.22: Timer synchronization (TIM15)*.

35.2 TIM15 main features

TIM15 includes the following features:

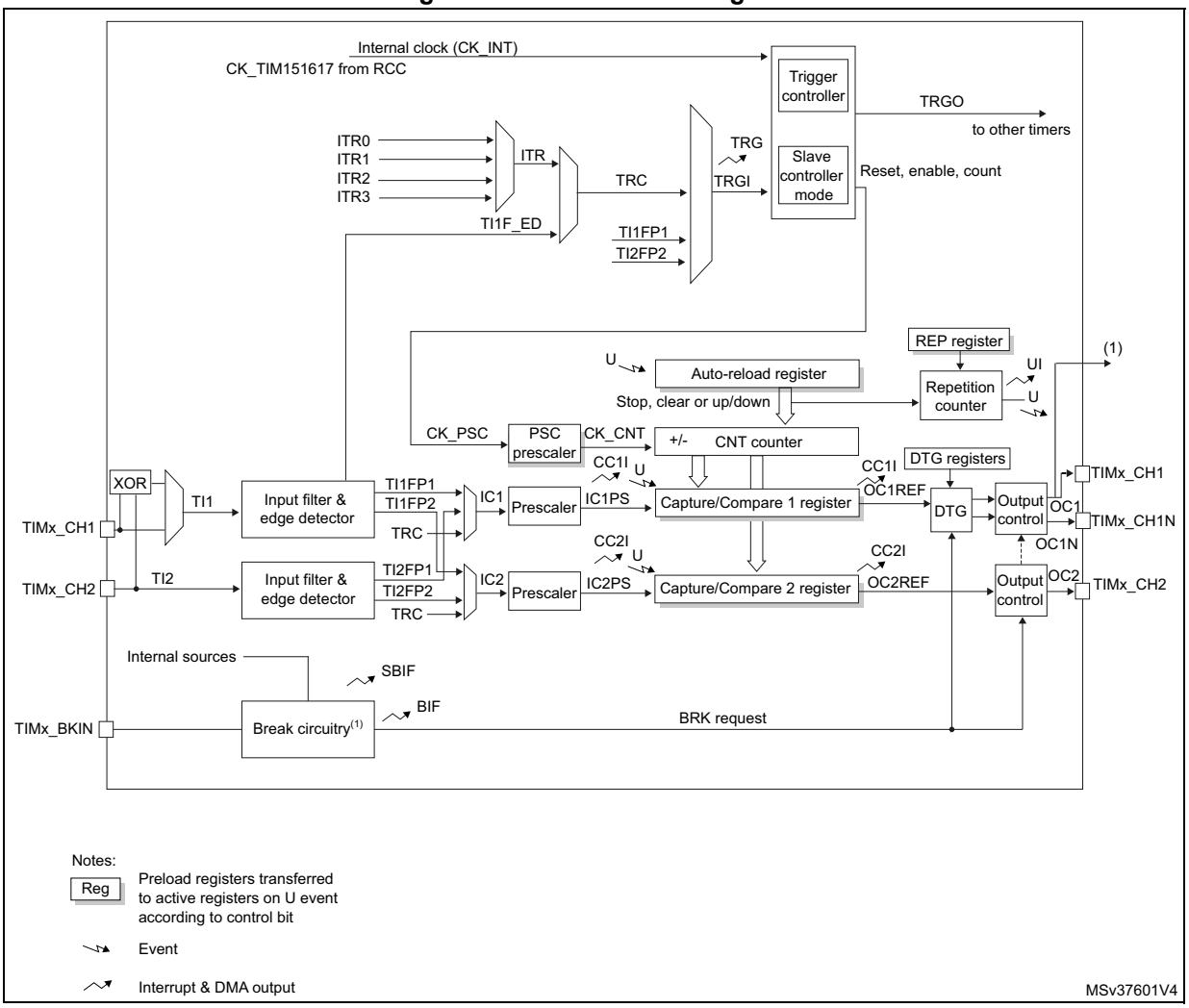
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary 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)

35.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

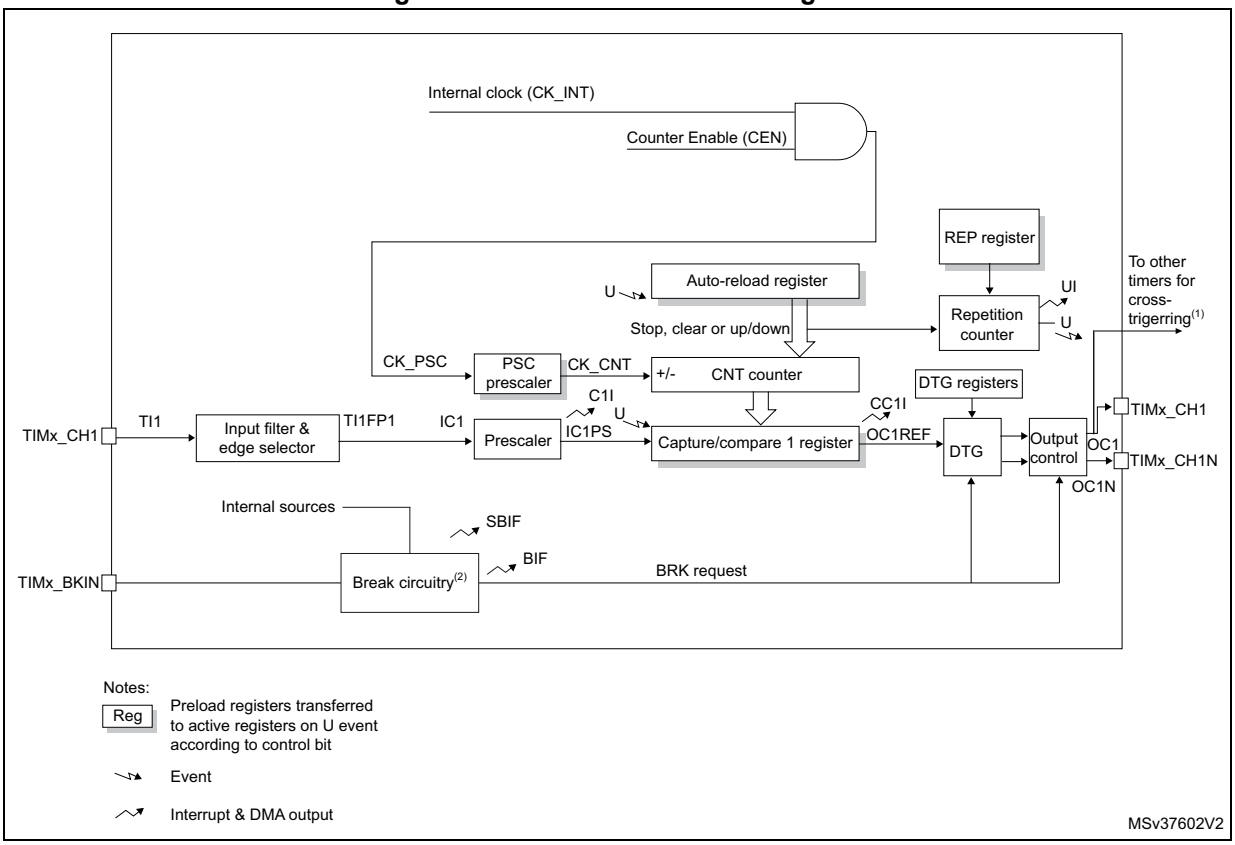
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs 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

Figure 339. TIM15 block diagram



1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 9.3.11: Clock security system (CSS)*
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M33 LOCKUP (Hardfault) output
   - COMP output

Figure 340. TIM16/TIM17 block diagram



1. This signal can be used as trigger for some slave timer, see *Section 35.4.23: Using timer output as trigger for other timers (TIM16/TIM17)*.
2. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 9.3.11: Clock security system (CSS)*
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M33 LOCKUP (Hardfault) output
   - COMP output

35.4 TIM15/TIM16/TIM17 functional description

35.4.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 341 and *Figure 342* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 341. Counter timing diagram with prescaler division change from 1 to 2

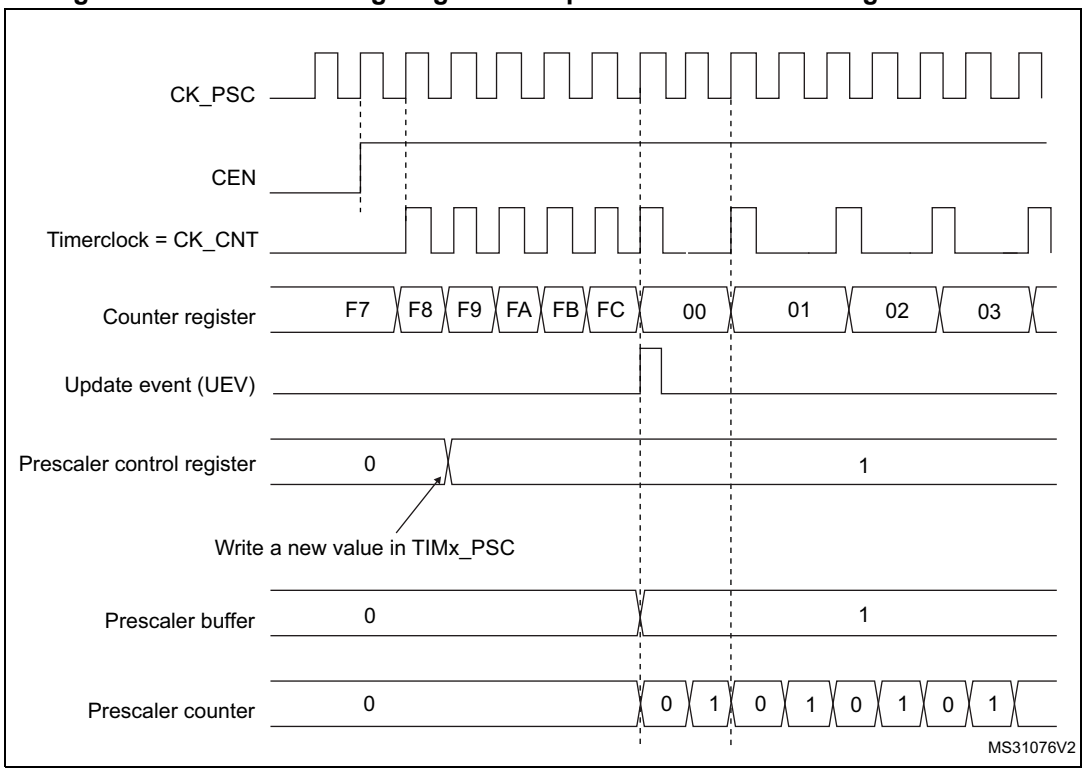
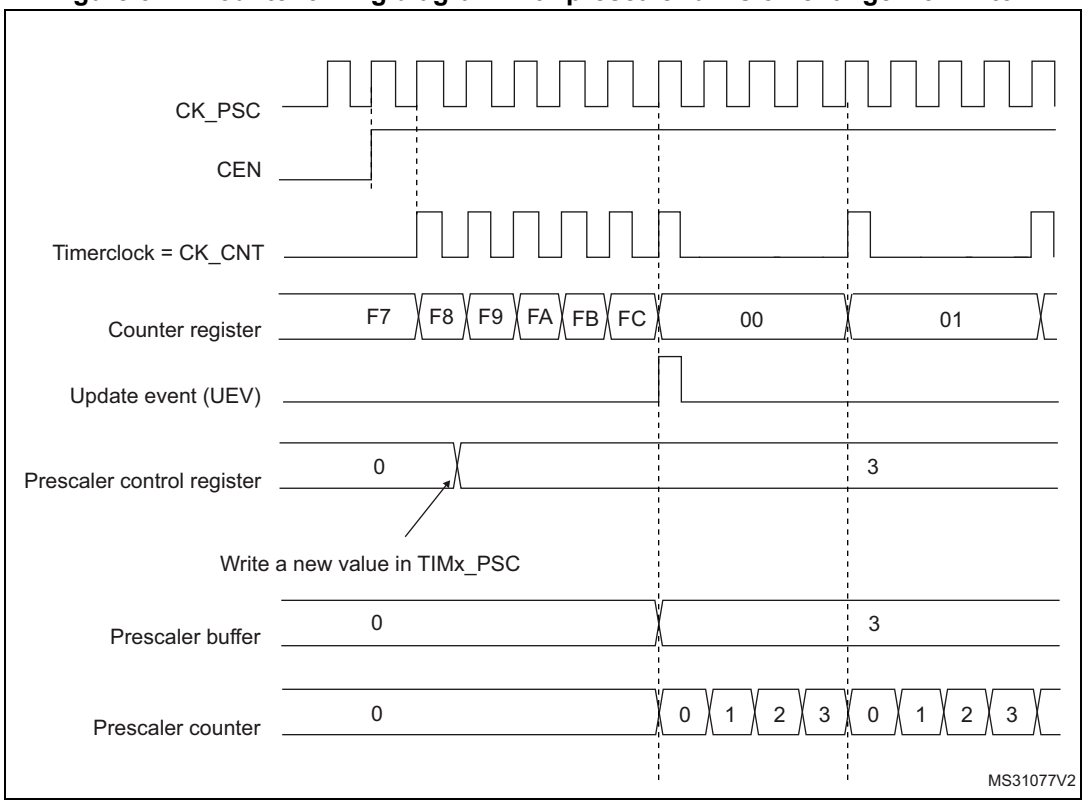


Figure 342. Counter timing diagram with prescaler division change from 1 to 4



35.4.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 343. Counter timing diagram, internal clock divided by 1

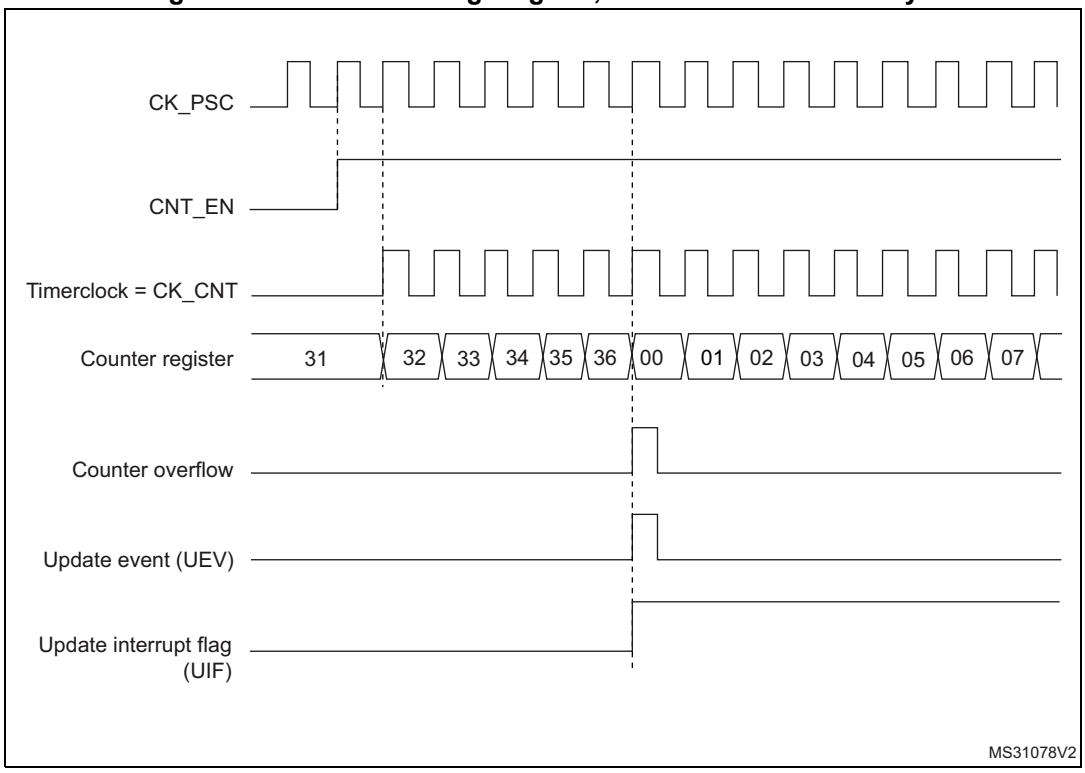


Figure 344. Counter timing diagram, internal clock divided by 2

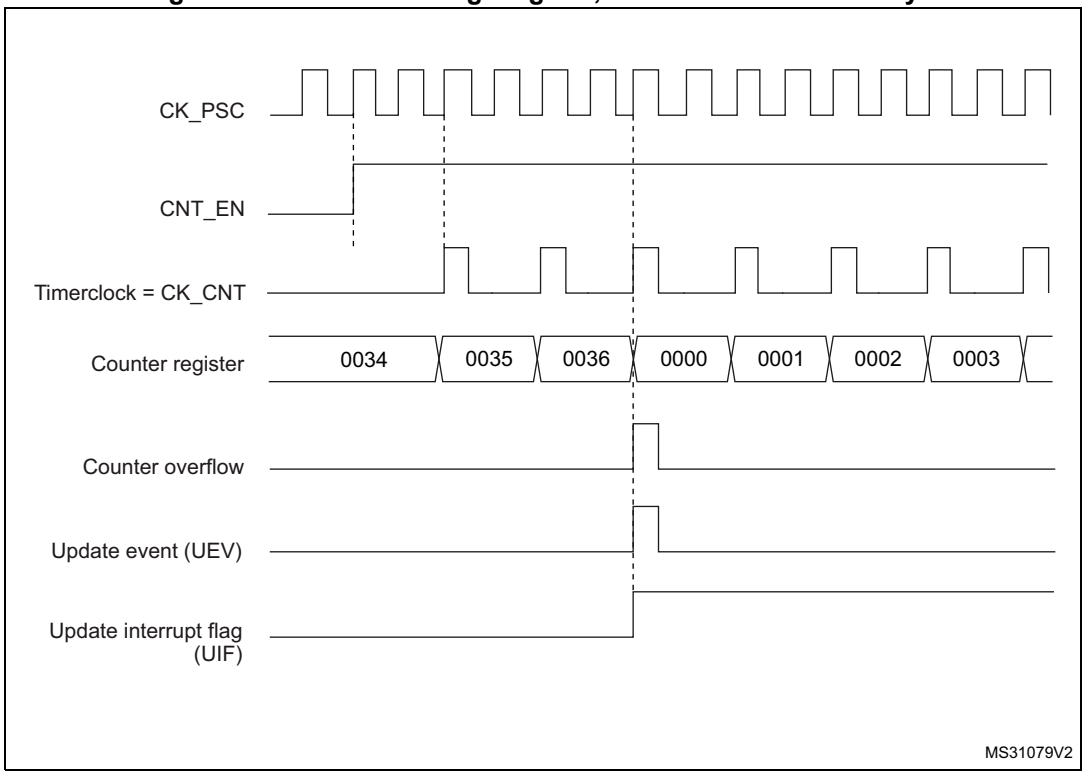


Figure 345. Counter timing diagram, internal clock divided by 4

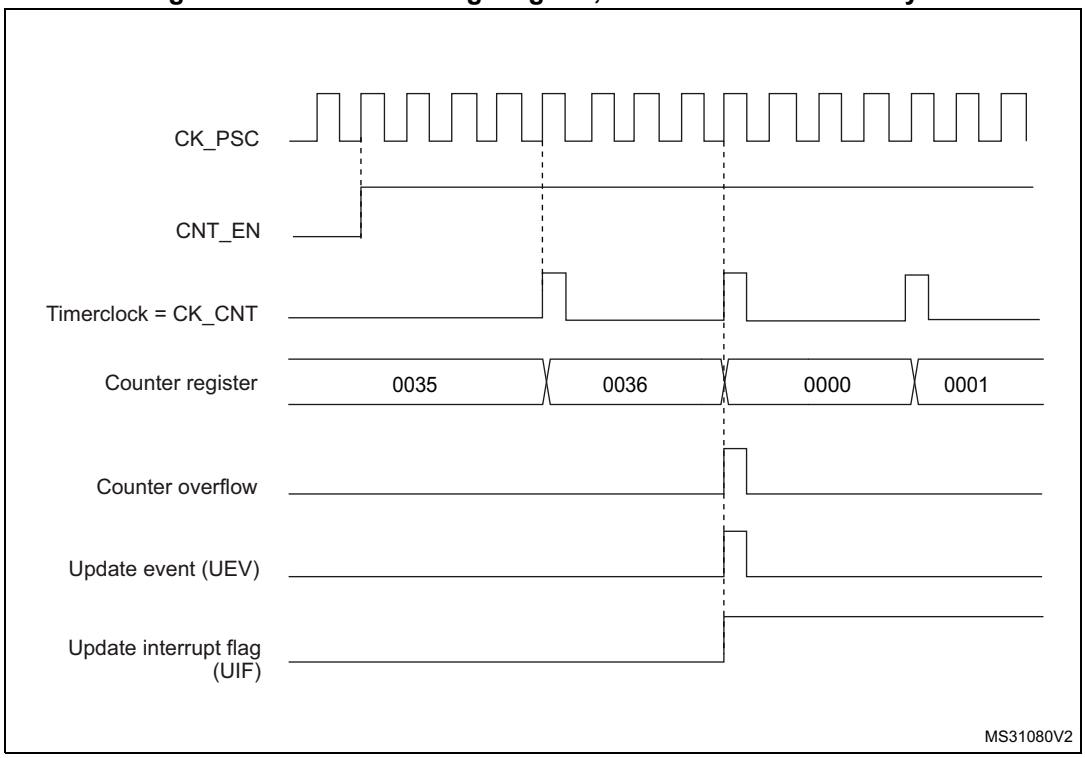


Figure 346. Counter timing diagram, internal clock divided by N

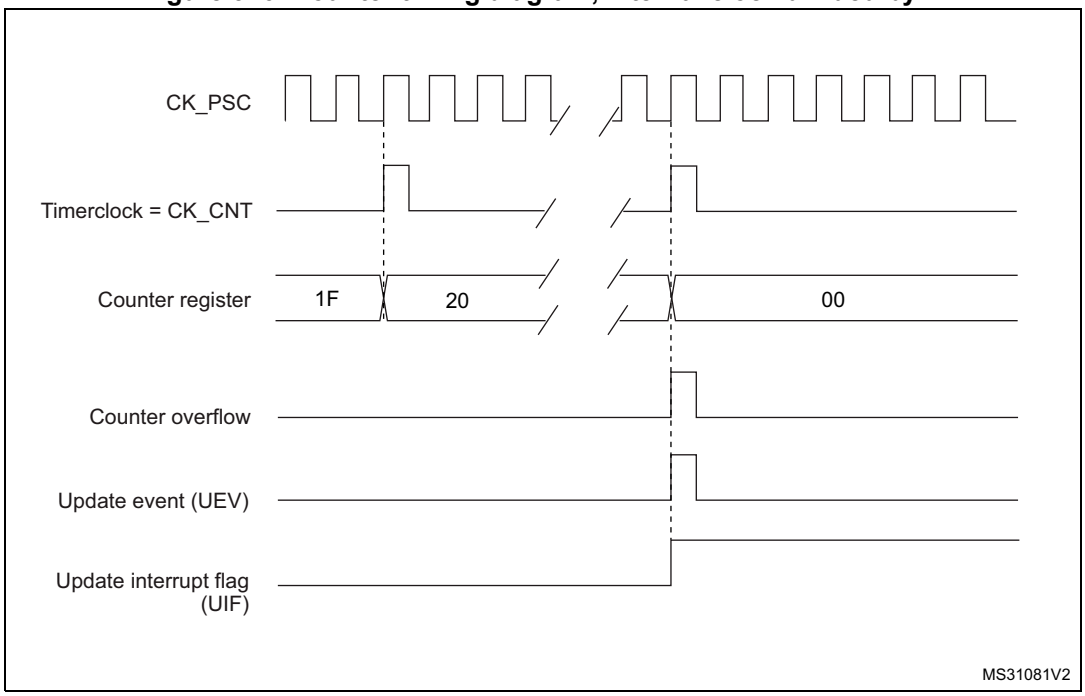


Figure 347. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

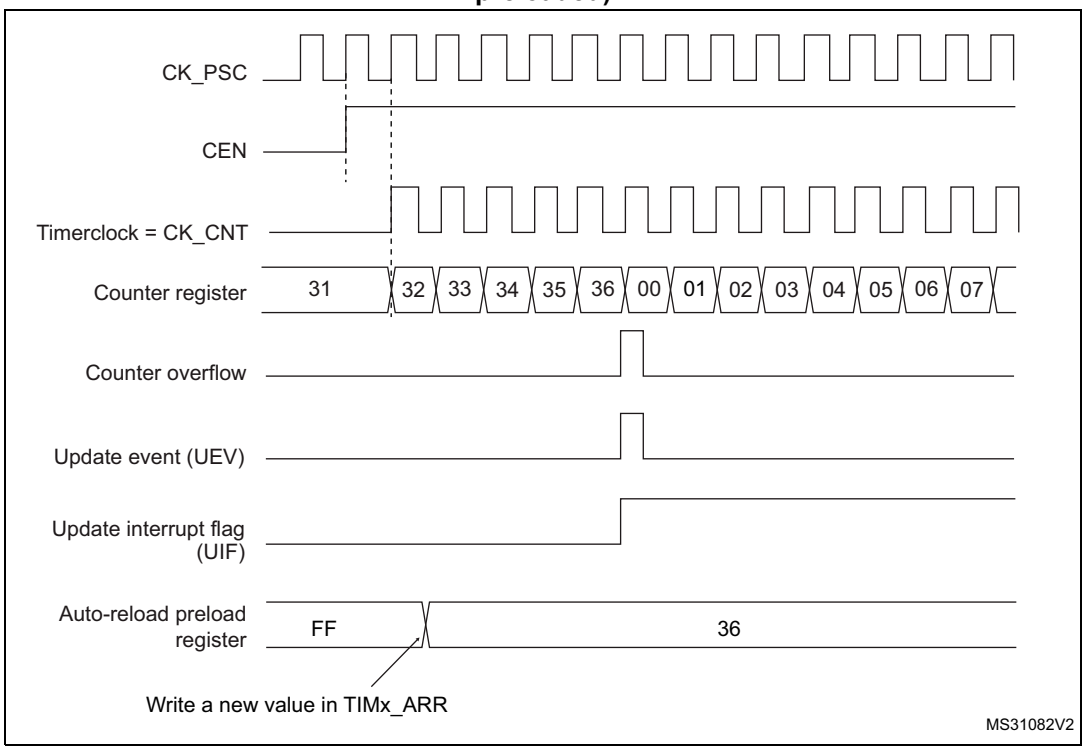
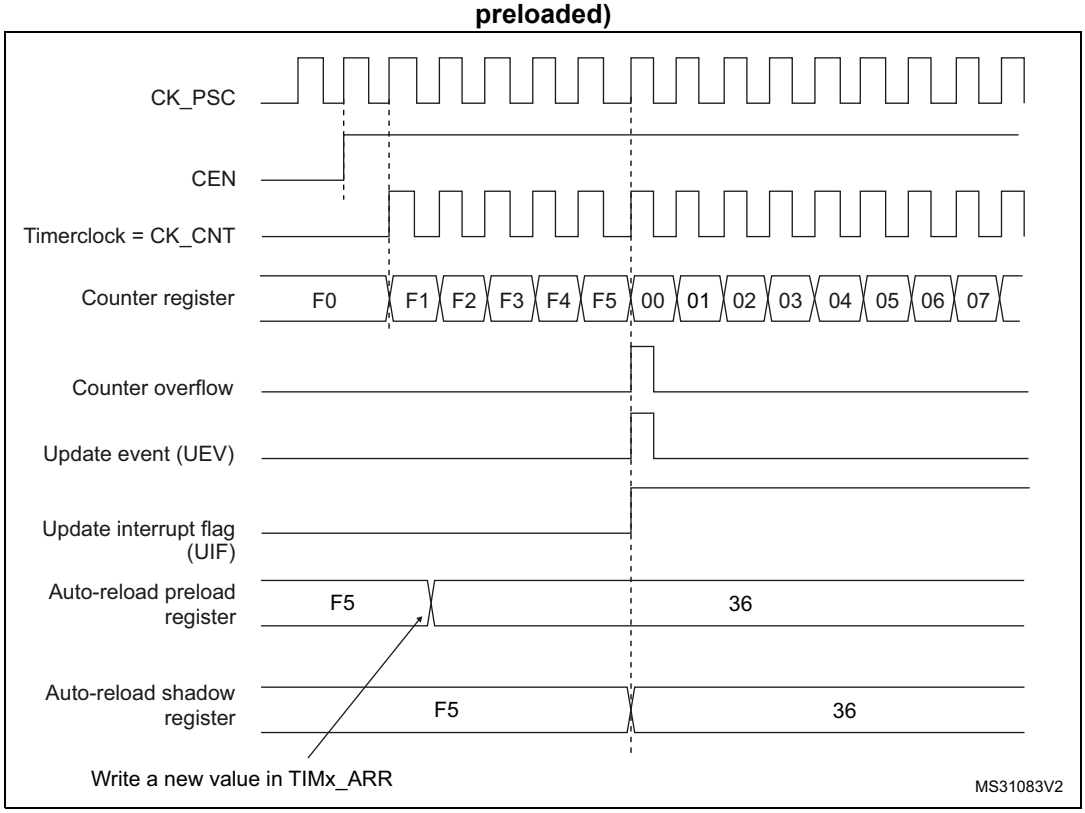


Figure 348. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



35.4.3 Repetition counter

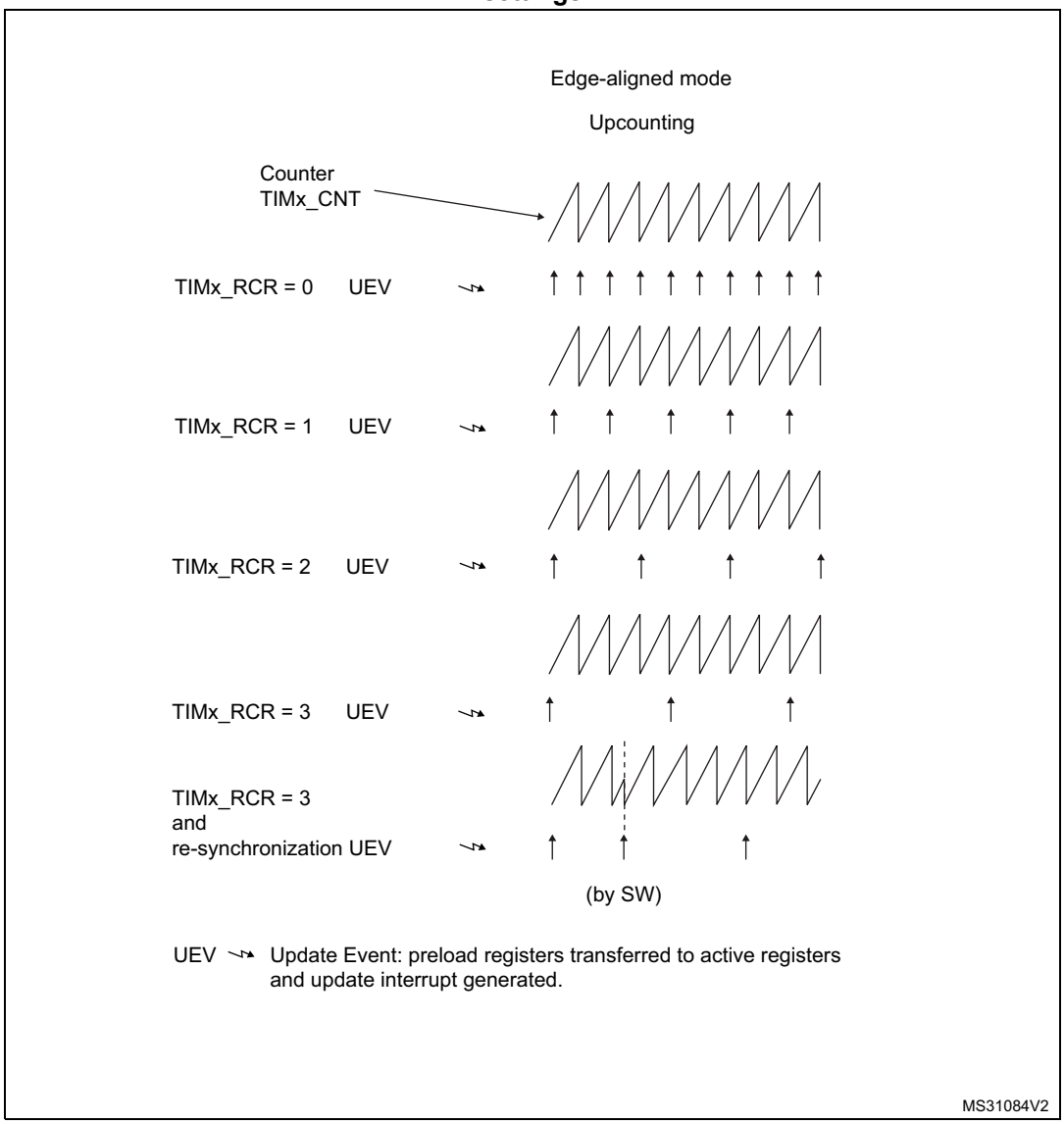
Section 35.4.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 349*). 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 349. Update rate examples depending on mode and TIMx_RCR register settings



35.4.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- Internal trigger inputs (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 as prescaler for another timer on page 1221* for more details.

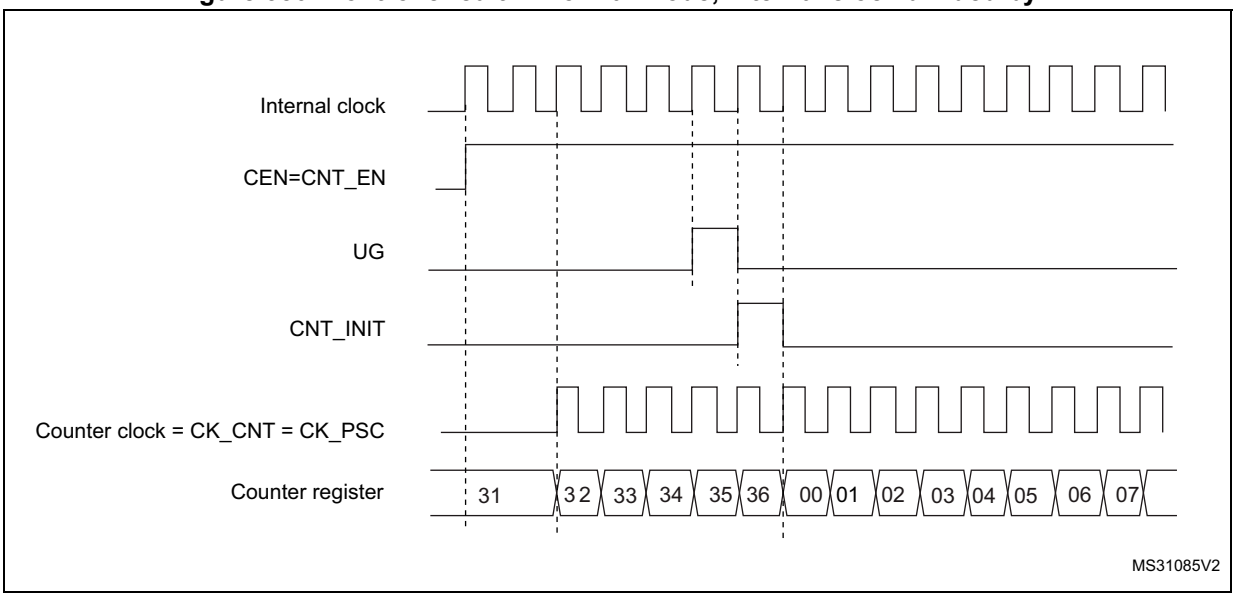
Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 350 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

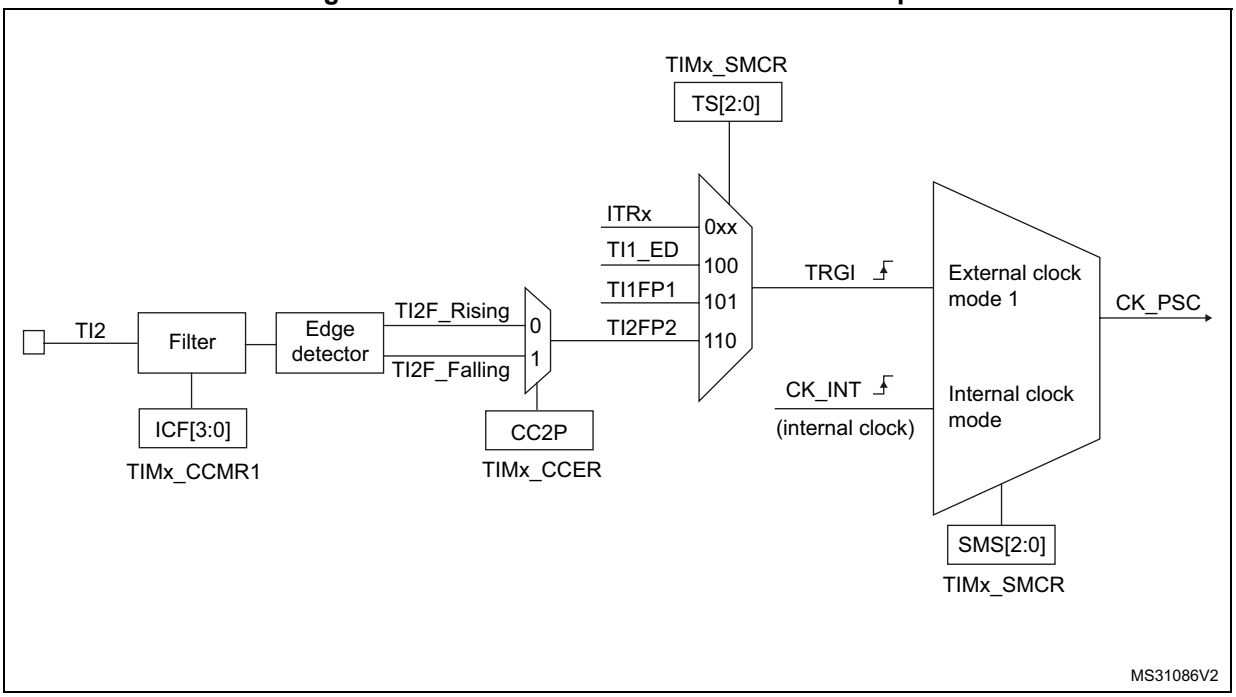
Figure 350. Control circuit in normal mode, internal clock divided by 1



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 351. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

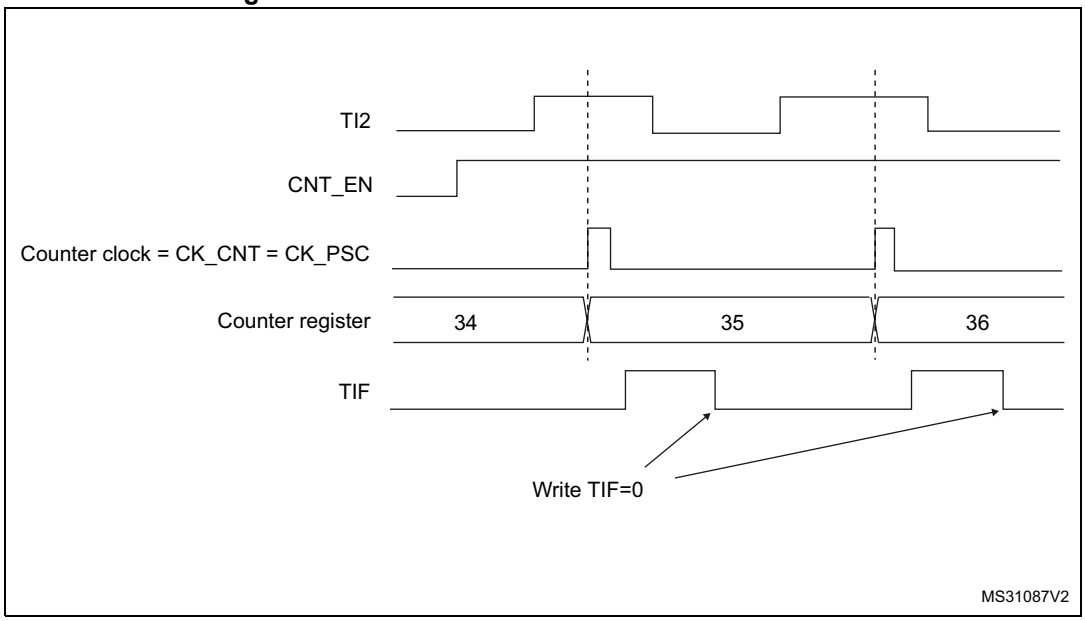
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 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 TI2 as the trigger input source by writing TS=110 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, so it does not need to be configured.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 352. Control circuit in external clock mode 1



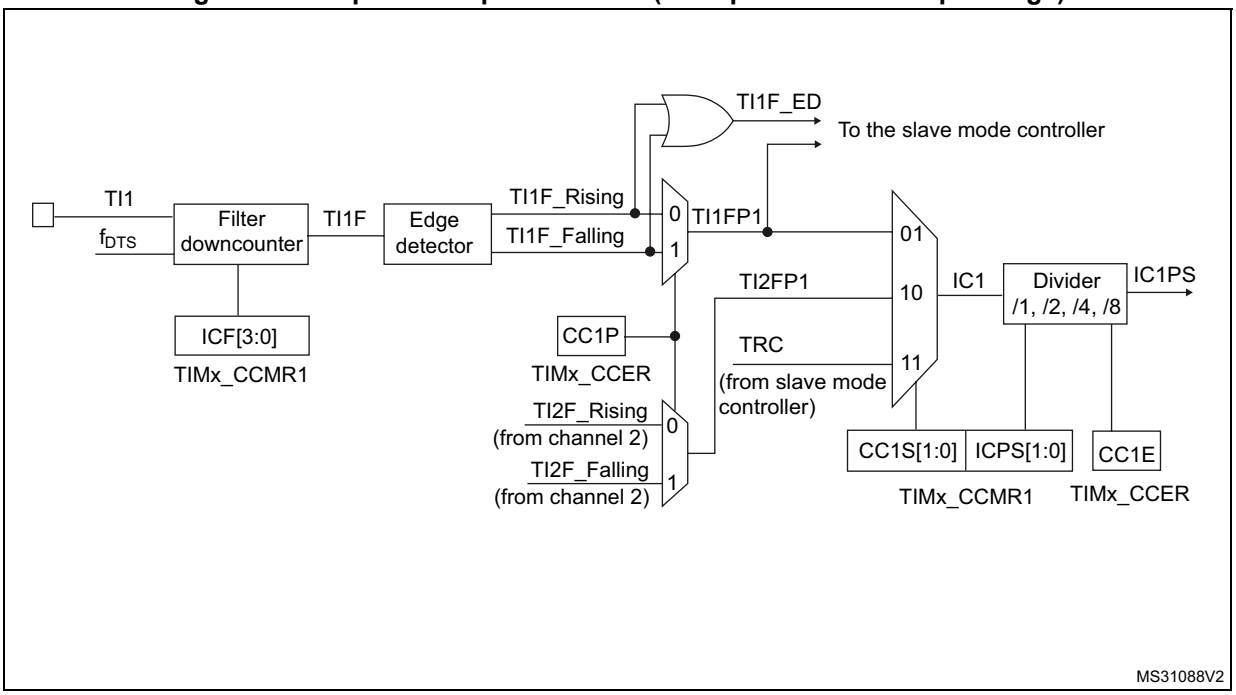
35.4.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 353 to *Figure 356* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 353. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 354. Capture/compare channel 1 main circuit

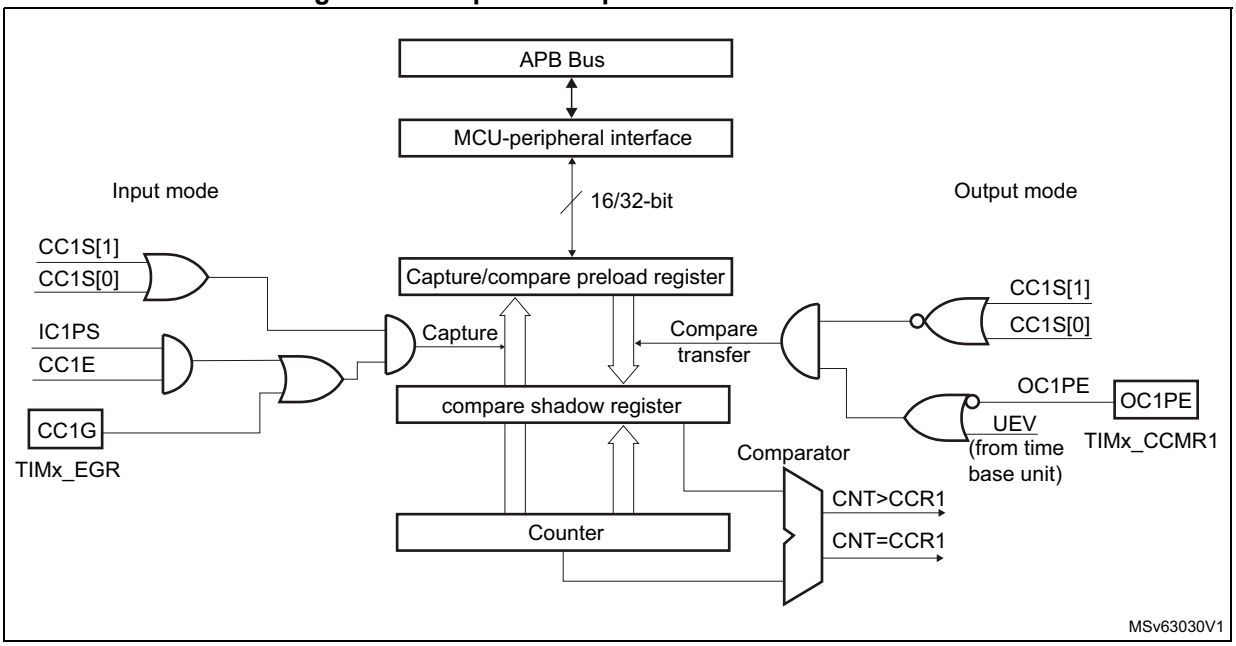


Figure 355. Output stage of capture/compare channel (channel 1)

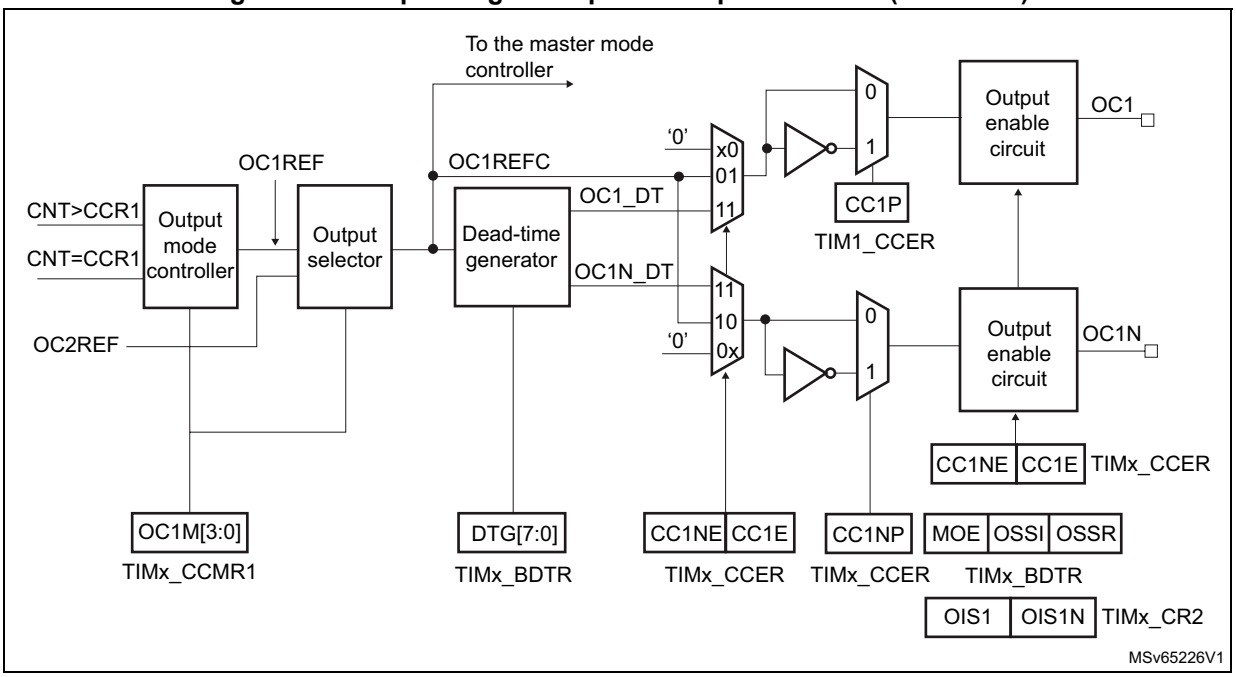
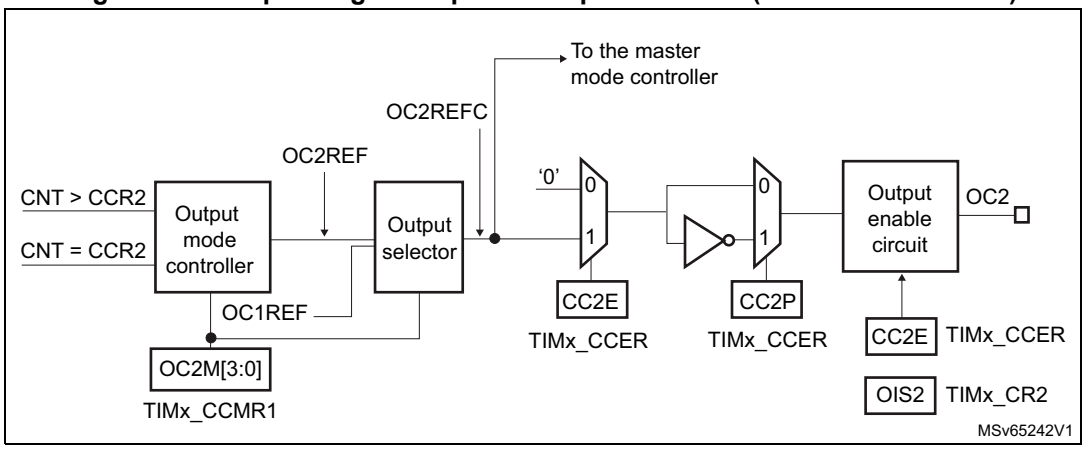


Figure 356. Output stage of capture/compare channel (channel 2 for TIM15)



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.

35.4.6 Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was

already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
3. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
4. 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).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. 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.

35.4.7 PWM input mode (only for TIM15)

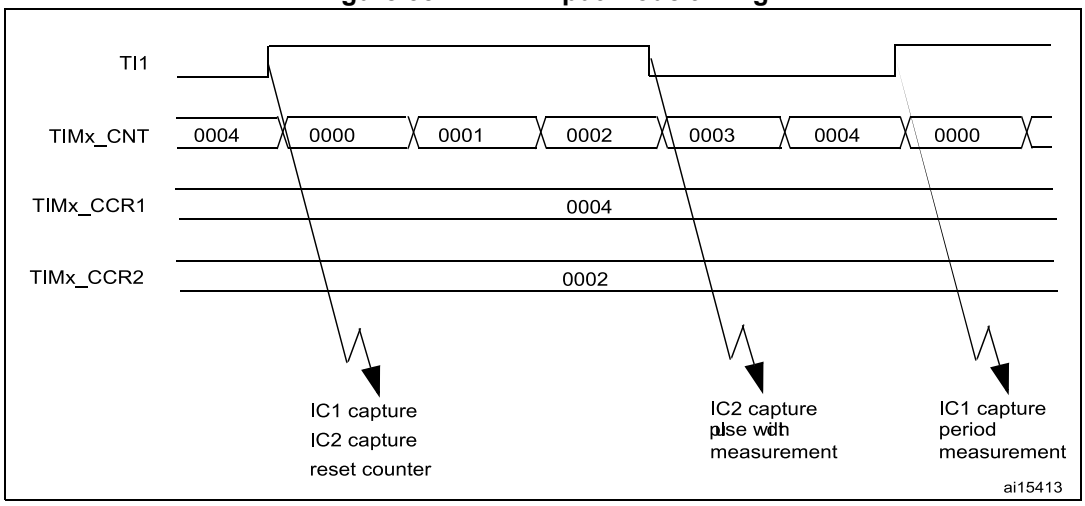
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to ‘10’ (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 357. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

35.4.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

35.4.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

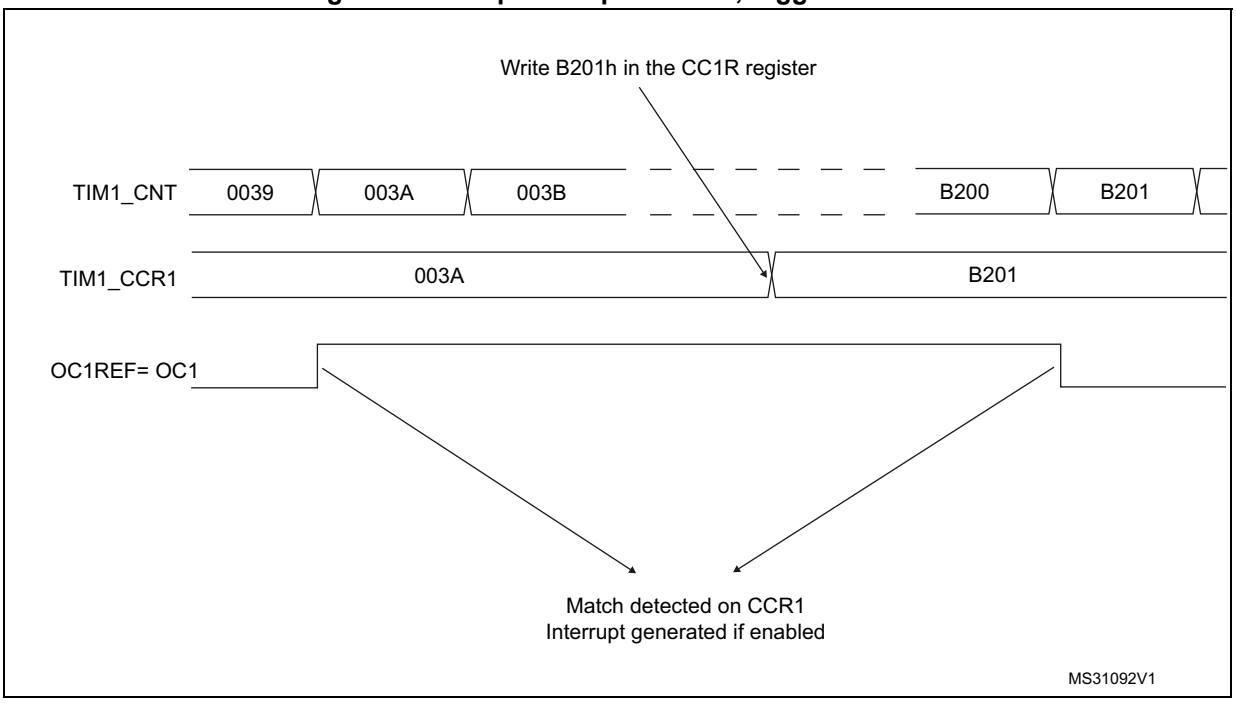
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 358*.

Figure 358. Output compare mode, toggle on OC1



35.4.10 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

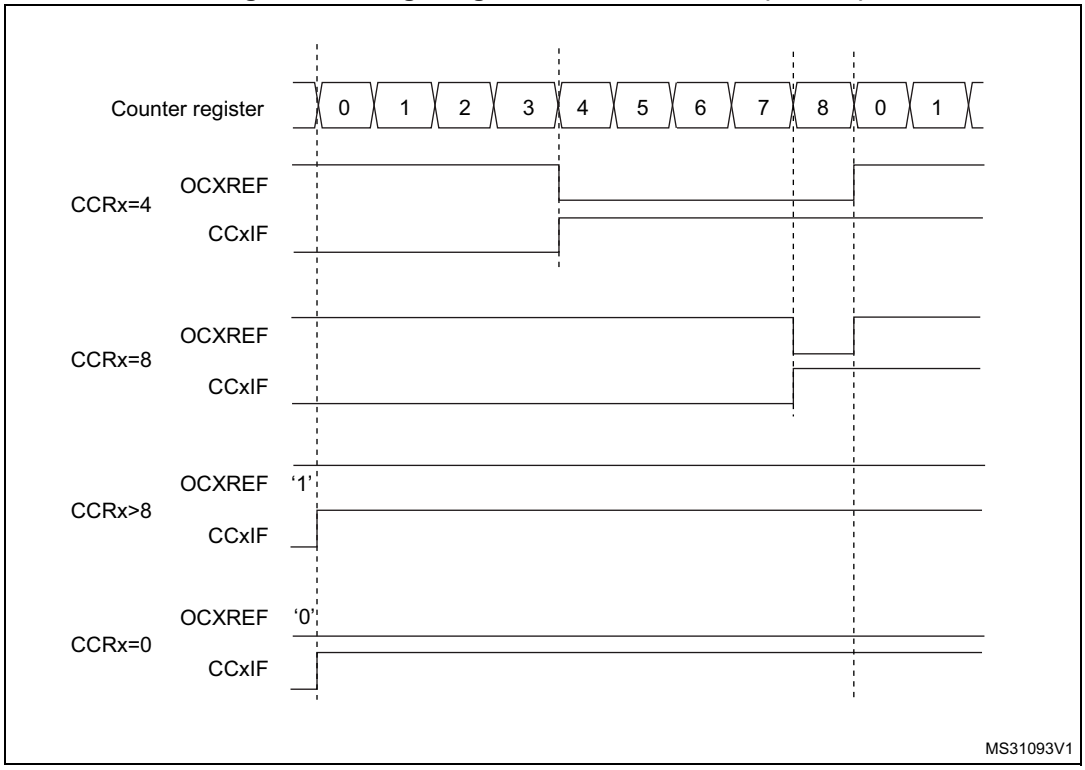
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode on page 1260*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at

'1'. If the compare value is 0 then OCxRef is held at '0'. *Figure 359* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 359. Edge-aligned PWM waveforms (ARR=8)



35.4.11 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, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

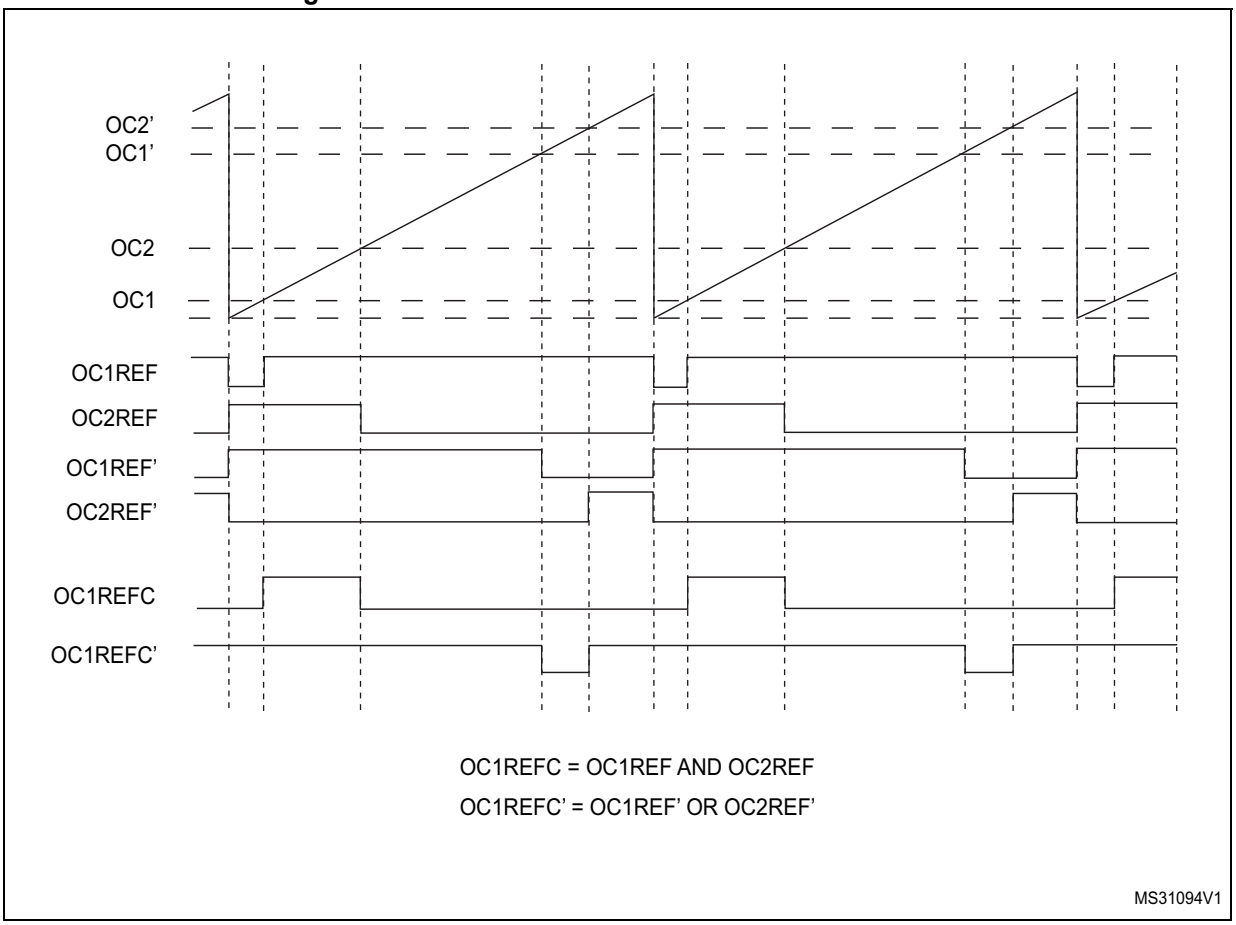
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 360 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,

Figure 360. Combined PWM mode on channel 1 and 2



35.4.12 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 OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 282: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) on page 1330* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 361. Complementary output with dead-time insertion.

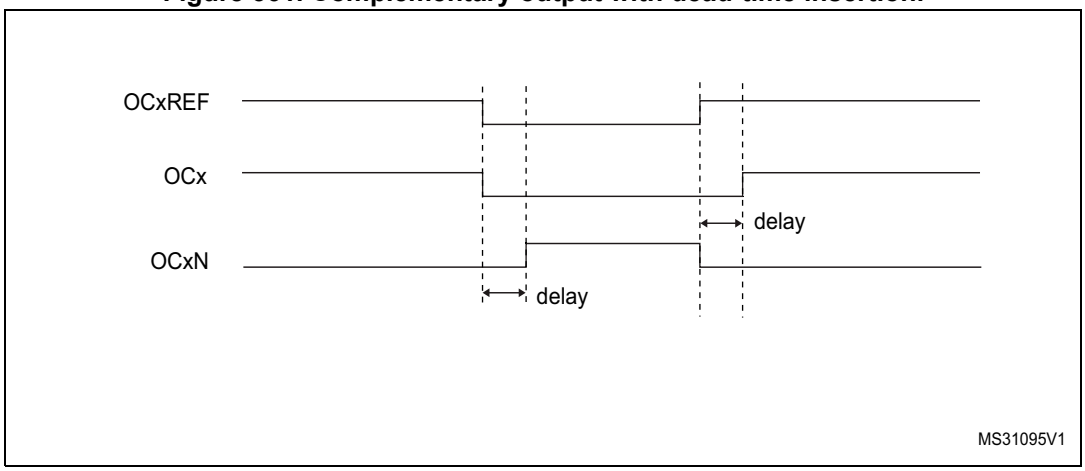


Figure 362. Dead-time waveforms with delay greater than the negative pulse.

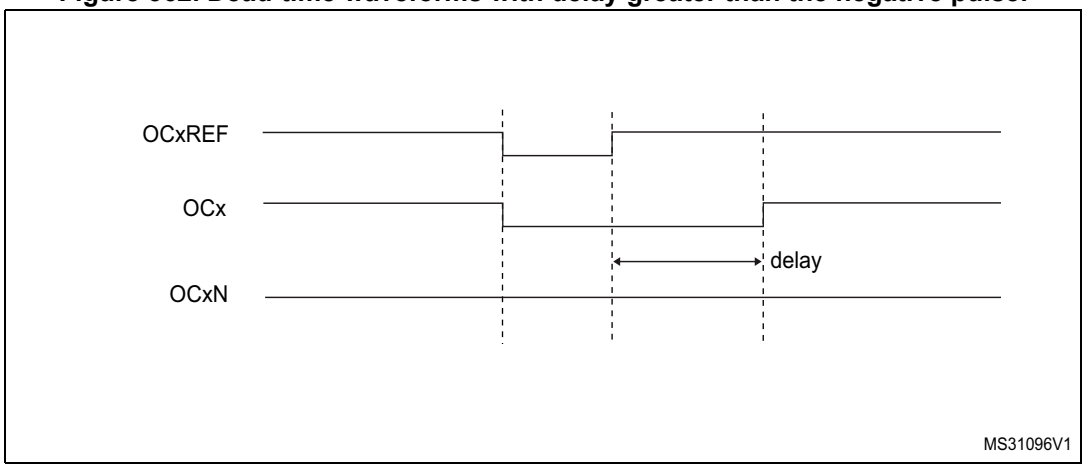
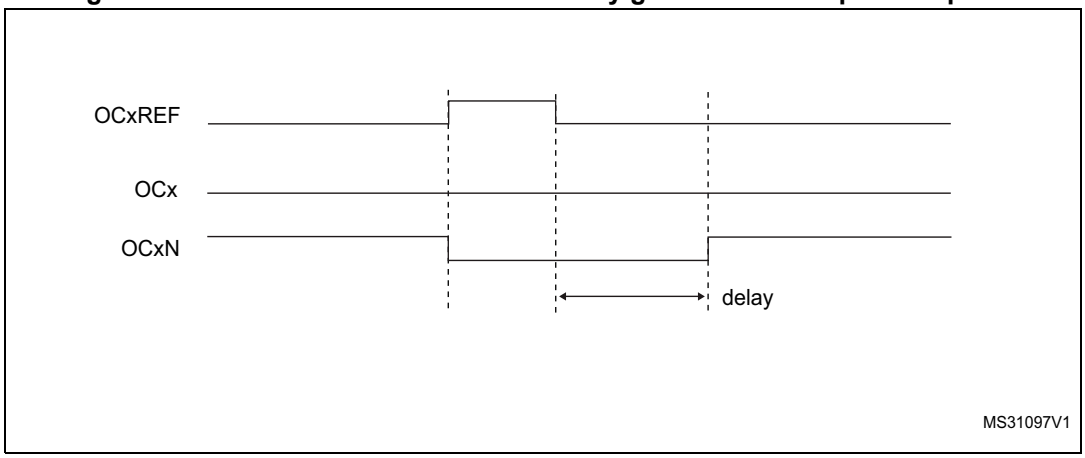


Figure 363. Dead-time waveforms with delay greater than the positive pulse.



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 35.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 1333* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

35.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

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 OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 280: Output control bits for complementary OCx and OCxN channels with break feature (TIM15) on page 1308* 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 can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_OR2 register.

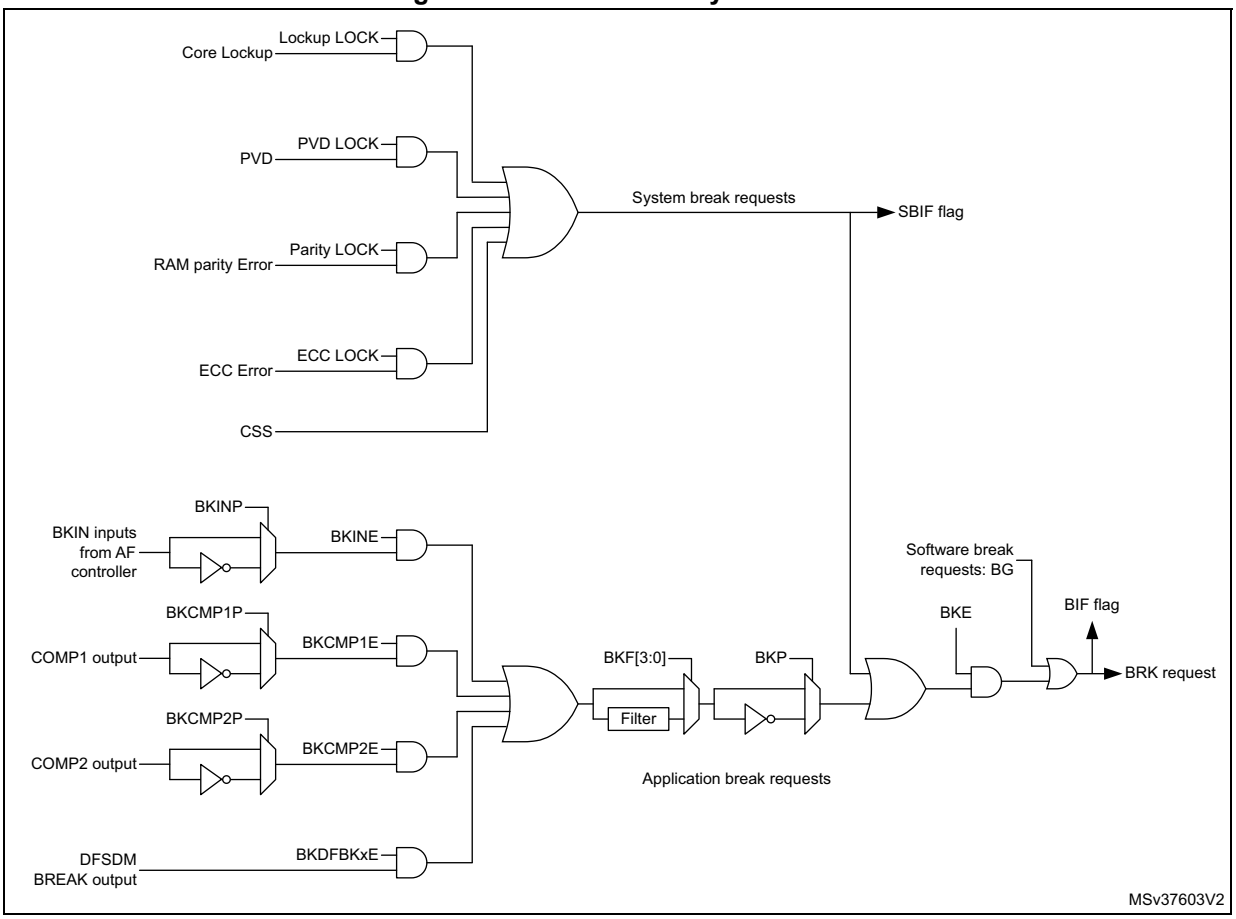
The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the Cortex®-M33 LOCKUP output
  - the PVD output
  - the SRAM parity error signal
  - a flash ECC error
  - a clock failure event generated by the CSS detector
  - the output from a comparator, with polarity selection and optional digital filtering
  - the analog watchdog output of the DFSDM1 peripheral

Break events can also be generated by software using BG bit in the TIMx_EGR register.

All sources are ORed before entering the timer BRK inputs, as per *Figure 364* below.

Figure 364. Break circuitry overview



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 AFIO controller (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 AFIO controller) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their

active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).

– If OSSI=0 then the timer releases the enable outputs (taken over by the AFIO controller which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

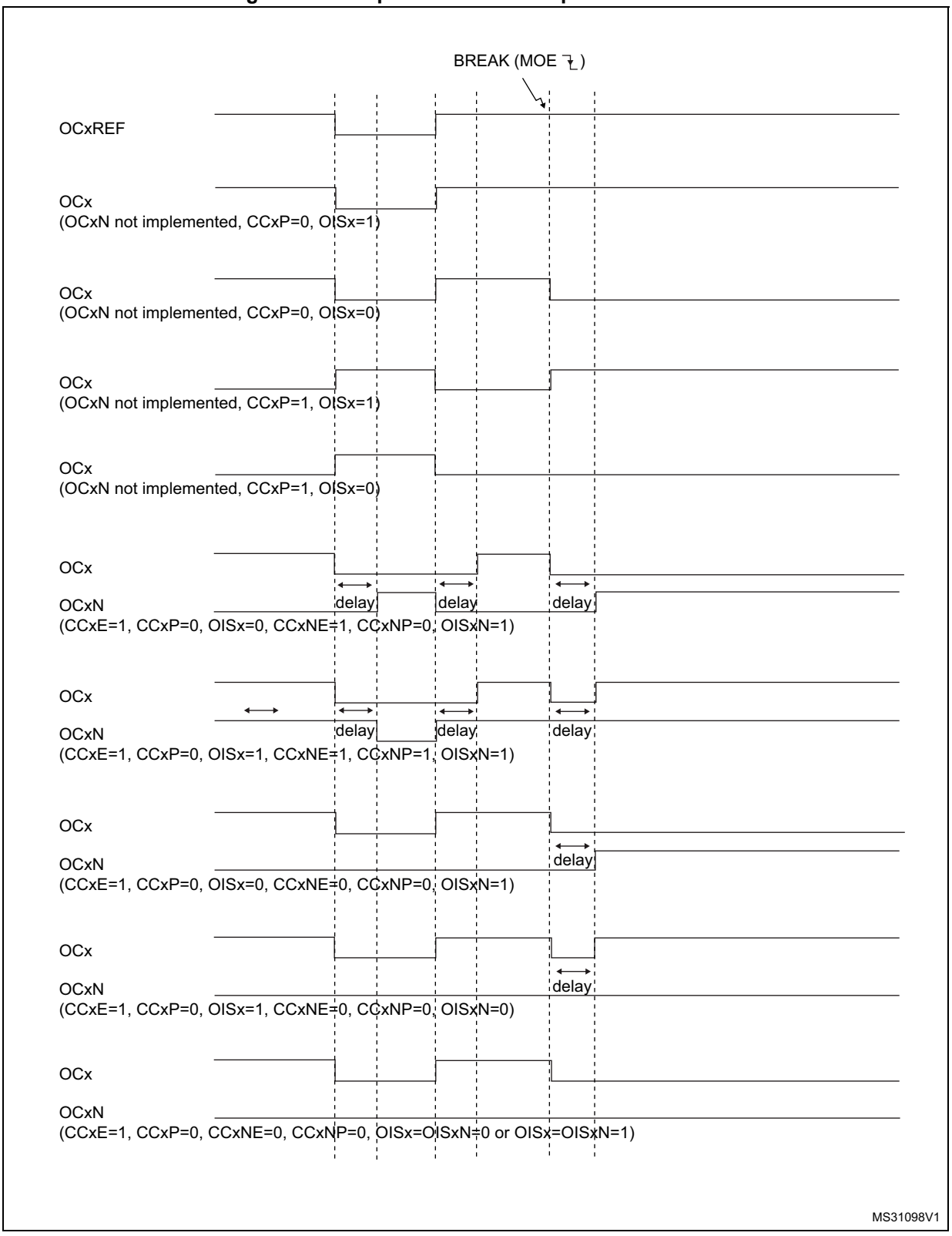
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 35.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 1333*. The LOCK bits can be written only once after an MCU reset.

The *Figure 365* shows an example of behavior of the outputs in response to a break.

Figure 365. Output behavior in response to a break



35.4.14 Bidirectional break inputs

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 366*.

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (BG) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 278*)

Table 278. Break protection disarming conditions

MOE	BKDIR	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

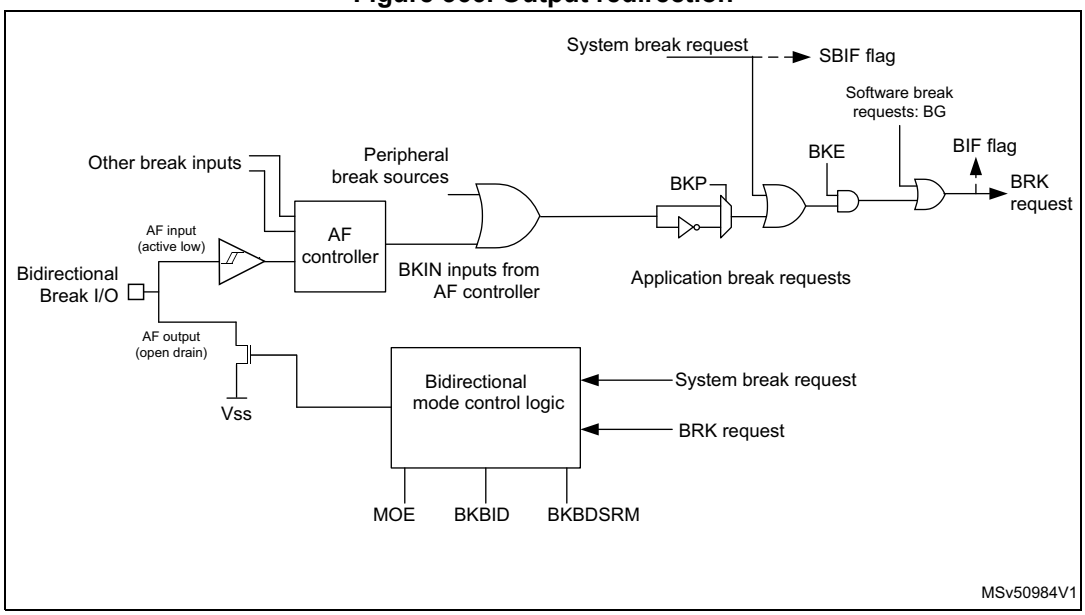
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 366. Output redirection



35.4.15 One-pulse mode

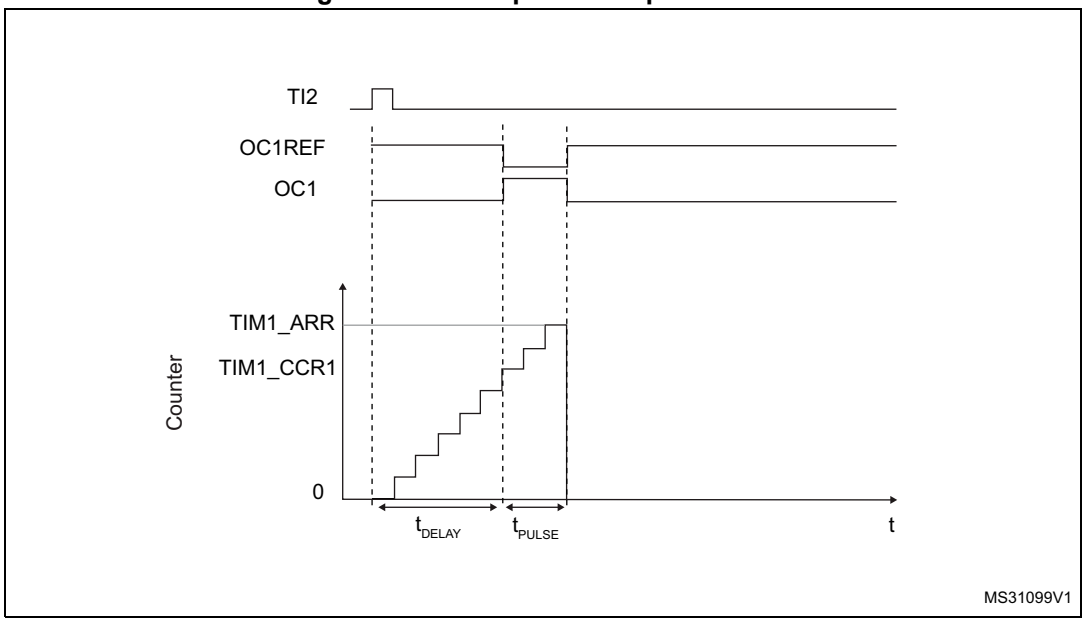
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 367. Example of one pulse mode



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

35.4.16 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 35.4.15*:

- 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

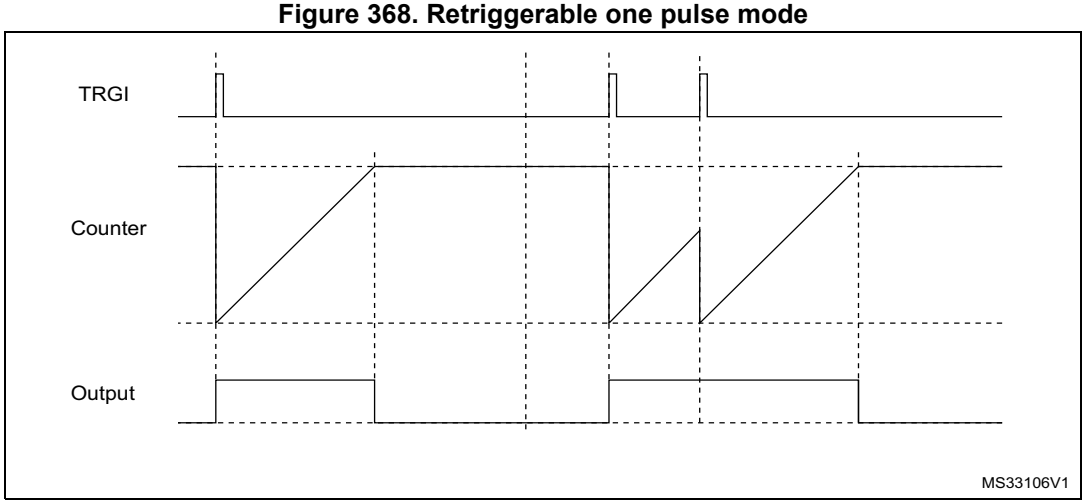
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retrigerrable OPM mode 1 or 2.

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 368. Retriggerable one pulse mode



35.4.17 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag, to be atomically read. In particular cases, it can ease the calculations by avoiding race conditions

caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

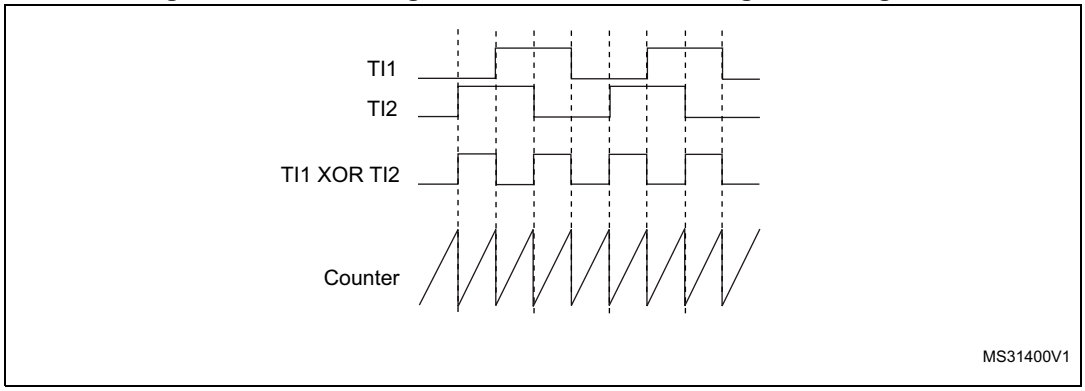
There is no latency between the assertions of the UIF and UIFCPY flags.

35.4.18 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 TIMx_CH1 and TIMx_CH2.

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 369*.

Figure 369. Measuring time interval between edges on 2 signals



35.4.19 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 and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

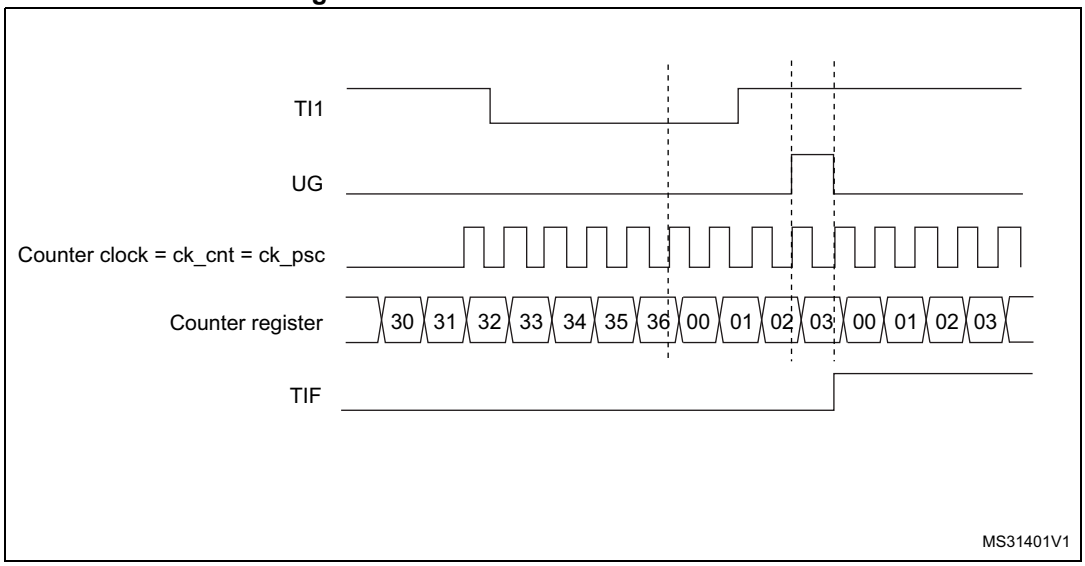
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in 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 TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 370. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

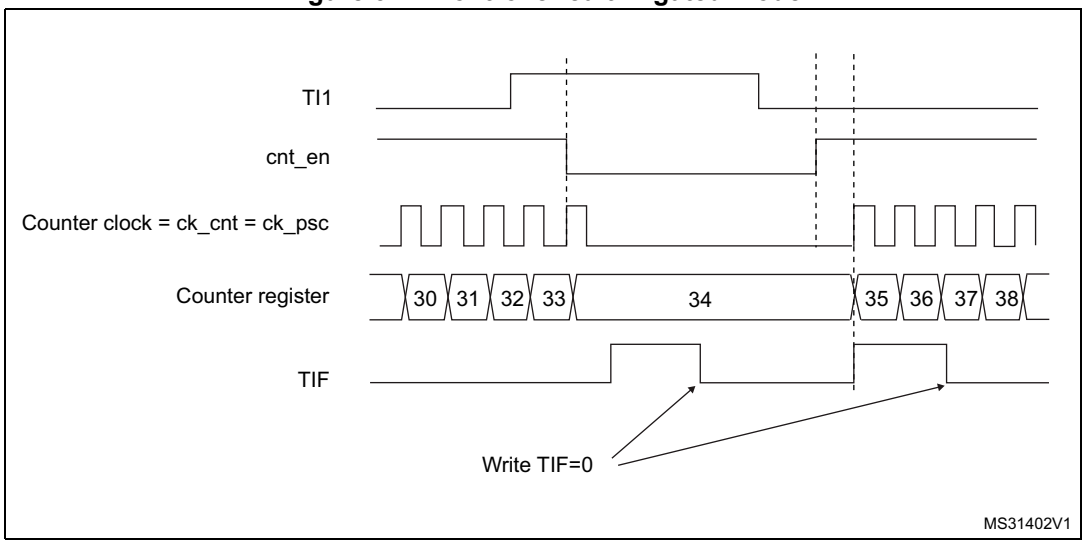
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = '0' in 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 TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 371. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

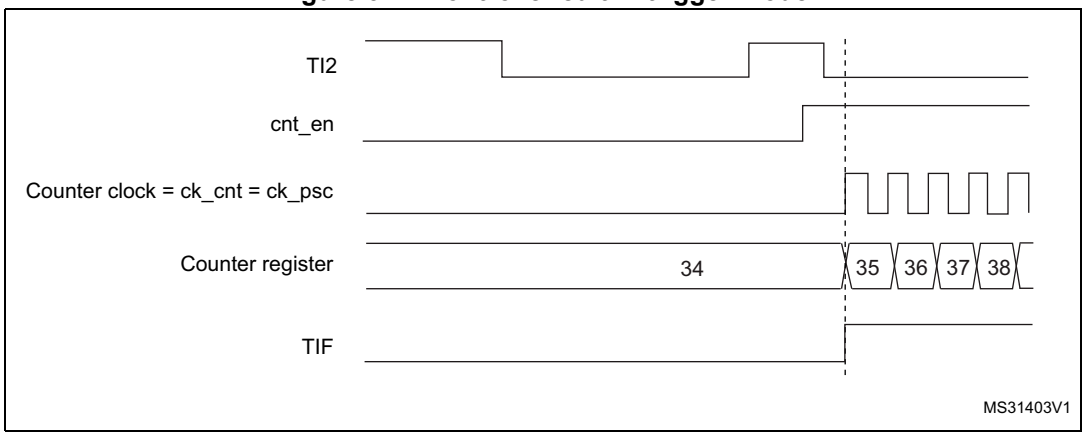
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. 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 TI2 as the input source by writing TS=110 in the TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 372. Control circuit in trigger mode



35.4.20 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

35.4.21 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.

35.4.22 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 34.3.19: Timer synchronization* for details.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.4.23 Using timer output as trigger for other timers (TIM16/TIM17)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger.

For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

35.4.24 Debug mode

When the microcontroller enters debug mode (Cortex®-M33 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module. For more details, refer to *Section : DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)*.

For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

35.5 TIM15 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).

35.5.1 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.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (TIx)
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 * t_{CK\_INT}$
10: $t_{DTS} = 4 * t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt 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.

35.5.2 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 (OC2 output)

0: OC2=0 when MOE=0

1: 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_BDTR register).

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 pins are connected to the 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 (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REFC signal is used as trigger output (TRGO).
101: **Compare** - OC2REFC signal is used as trigger output (TRGO).

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.

35.5.3 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.	Res.	Res.	Res.	Res.	Res.	SMS[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.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:17 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 (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bit field selects the trigger input to be used to synchronize the counter.

000: Internal Trigger 0 (ITR0)
001: Internal Trigger 1 (ITR1)
010: Internal Trigger 2 (ITR2)
011: Internal Trigger 3 (ITR3)
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)

See *Table 279: TIMx Internal trigger connection on page 1298* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 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 (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 (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Other codes: reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=’100’). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 279. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM15	TIM1	TIM3	TIM16 OC1	TIM17 OC1

35.5.4 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.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

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

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

35.5.5 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 TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, 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 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.

– When CNT is reinitialized by a trigger event (refer to *Section 35.5.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

35.5.6 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 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 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 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).

35.5.7 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 in input and in output mode.

Input capture mode:

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

35.5.8 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 in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

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, IC2 is mapped on TI2.
10: CC2 channel is configured as input, IC2 is mapped on TI1.
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Reserved,

1111: 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, 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 that have 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.

The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

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 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, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

35.5.9 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: OC1N active high

1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 280* for details.

Table 280. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>OCx output state</th><th>OCxN output state</th></tr>
<tr><td rowspan="7">1</td><td rowspan="7">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
OCx=0
OCxN=0, OCxN_EN=0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCx=0</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF XOR CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCxN=0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
OCx=CCxP</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF xor CCxP,
OCx_EN=1</td><td>Off-State (output enabled with inactive state)
OCxN=CCxNP, OCxN_EN=1</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z)</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: OCx=CCxP, OCxN=CCxNP
Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

35.5.10 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 TIMx_ISR register.

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

Bits 15:0 **CNT[15:0]**: Counter value

35.5.11 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 (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

35.5.12 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 35.4.1: Time-base unit on page 1258* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

35.5.13 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 value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

35.5.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

35.5.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_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 signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

35.5.16 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.	Res.	Res.	Res.	Res.
			rw		rw										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input BRK in input mode

1: Break input BRK in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input BRK is armed
1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)

See OC/OCN enable description for more details (*Section 35.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1306*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low
1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled
1; Break inputs (BRK and CCS clock failure event) enabled

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 that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 35.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1306*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO 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 on channels configured as outputs.

See OC/OCN enable description for more details (*Section 35.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1306*).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)

1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx => DT = DTG[7:0] x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x => DT = (64+DTG[5:0]) x t_{dtg} with $t_{dtg} = 2 \times t_{DTS}$

DTG[7:5] = 110 => DT = (32+DTG[4:0]) x t_{dtg} with $t_{dtg} = 8 \times t_{DTS}$

DTG[7:5] = 111 => DT = (32+DTG[4:0]) x t_{dtg} with $t_{dtg} = 16 \times t_{DTS}$

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

35.5.17 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit 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).
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,
...

35.5.18 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

35.5.19 TIM15 option register 1 (TIM15_OR1)

Address offset: 0x50

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	ENCODER_ MODE[1:0]		TI1_ RMP
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:1 **ENCODER_MODE[1:0]**: Encoder mode
00: No redirection
01:TIM2 IC1 and TIM2 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively
10:TIM3 IC1 and TIM3 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively
11:TIM4 IC1 and TIM4 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively

Bit 0 **TI1_RMP**: Input capture 1 remap
0: TIM15 input capture 1 is connected to I/O
1: TIM15 input capture 1 is connected to LSE

35.5.20 TIM15 option register 2 (TIM15_OR2)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCM P2P	BKCM P1P	BKINP	BKDF1 BK0E	Res.	Res.	Res.	Res.	Res.	BKCM P2E	BKCM P1E	BKINE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active low
1: COMP2 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK0E**: BRK dfsdm1_break[0] enable

This bit enables the dfsdm1_break[0] for the timer's BRK input. dfsdm1_break[0] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[0]input disabled

1: dfsdm1_break[0]input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

35.5.21 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 281. 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.	Res.	Res.	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
0x0C	**TIM15_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	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** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	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
	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
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

Table 281. 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
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.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**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.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM15_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIM15_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DTG[7:0]							
	Reset value				0		0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**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
0x4C	**TIM15_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM15_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.	ENCODER_MODE[1:0]		TI1_RMP
	Reset value																														0	0	0

Table 281. 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_OR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK0E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																					0	0	0	0						0	0	1

Refer to *Section 2.3 on page 87* for the register boundary addresses.

35.6 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

35.6.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (TIx),
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 * t_{CK\_INT}$
10: $t_{DTS} = 4 * t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

35.6.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

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

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

35.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

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

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

35.6.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

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

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

35.6.5 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).

35.6.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

35.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
									rw	rw	rw	rw	rw	rw	rw

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

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

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

All other values: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

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 of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

35.6.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high
1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 282* for details.

Table 282. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

35.6.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

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

Bits 15:0 **CNT[15:0]**: Counter value

35.6.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

35.6.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 35.4.1: Time-base unit on page 1258* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

35.6.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

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

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

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

35.6.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

35.6.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw		rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input BRK in input mode
1: Break input BRK in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)
See OC/OCN enable description for more details (*Section 35.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1328*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1; Break inputs (BRK and CCS clock failure event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 35.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1328*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO 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 on channels configured as outputs.
See OC/OCN enable description for more details (*Section 35.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1328*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0] x t_{dtg} with $t_{dtg} = t_{DTS}$
DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x t_{dtg} with $t_{dtg} = 2 \times t_{DTS}$
DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t_{dtg} with $t_{dtg} = 8 \times t_{DTS}$
DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t_{dtg} with $t_{dtg} = 16 \times t_{DTS}$
Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

35.6.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

35.6.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

35.6.17 TIM16 option register 1 (TIM16_OR1)

Address offset: 0x50

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits1:0 **TI1_RMP[1:0]**: Input capture 1 remap
00: TIM16 input capture 1 is connected to I/O
01: TIM16 input capture 1 is connected to LSI
10: TIM16 input capture 1 is connected to LSE
11: TIM16 input capture 1 is connected to RTC wakeup interrupt

35.6.18 TIM16 option register 2 (TIM16_OR2)

Address offset: 0x60

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK1E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active low
1: COMP2 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK1E**: BRK dfsdm1_break[1] enable

This bit enables the dfsdm1_break[1] for the timer's BRK input. dfsdm1_break[1] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[1] input disabled

1: dfsdm1_break[1] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

35.6.19 TIM17 option register 1 (TIM17_OR1)

Address offset: 0x50

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits1:0 **TI1_RMP[1:0]**: Input capture 1 remap
00: TIM17 input capture 1 is connected to I/O
01: TIM17 input capture 1 is connected to MSI
10: TIM17 input capture 1 is connected to HSE/32
11: TIM17 input capture 1 is connected to MCO

35.6.20 TIM17 option register 2 (TIM17_OR2)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active low
1: COMP2 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active low
1: BKIN input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK2E**: BRK dfsdm1_break[2] enable
This bit enables the dfsdm1_break[2] for the timer's BRK input. dfsdm1_break[2] output is 'ORed' with the other BRK sources.
0: dfsdm1_break[2] input disabled
1: dfsdm1_break[2] input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

35.6.21 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 283. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0										0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 283. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	**TIMx_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIMx_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DTG[7:0]							
	Reset value				0		0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM16_OR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP [1:0]	
	Reset value																															0	0
0x60	**TIM16_OR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK1E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																					0	0	0	0						0	0	1
0x50	**TIM17_OR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP [1:0]	
	Reset value																															0	0
0x60	**TIM17_OR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																					0	0	0	0						0	0	1

Refer to *Section 2.3 on page 87* for the register boundary addresses.

36 Basic timers (TIM6/TIM7)

36.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

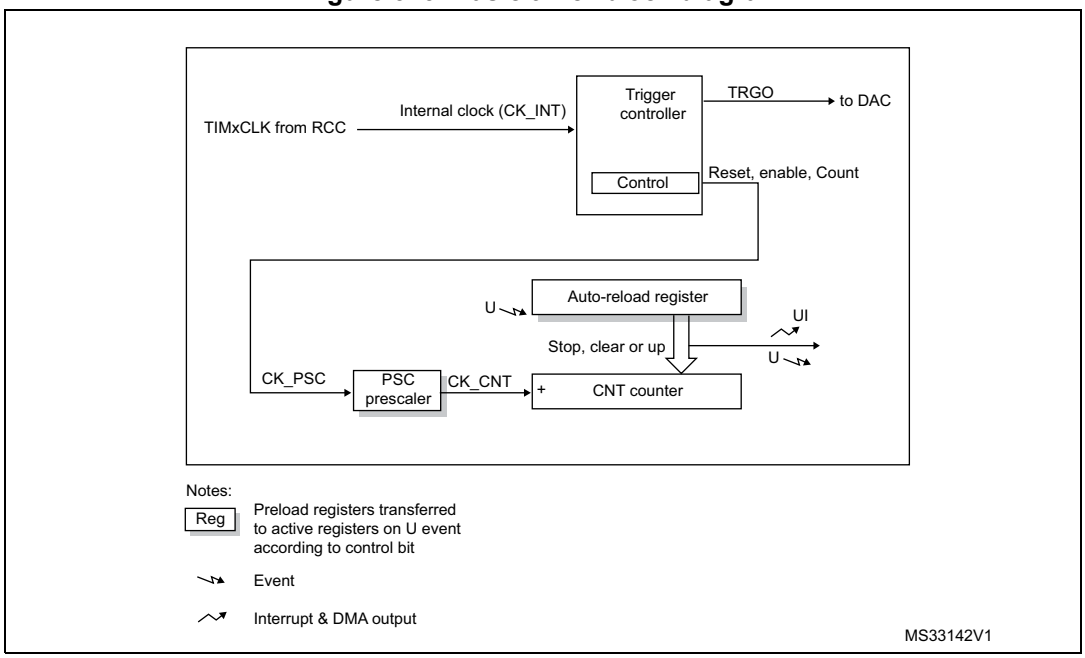
The timers are completely independent, and do not share any resources.

36.2 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

Figure 373. Basic timer block diagram



36.3 TIM6/TIM7 functional description

36.3.1 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 CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

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 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 374 and *Figure 375* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 374. Counter timing diagram with prescaler division change from 1 to 2

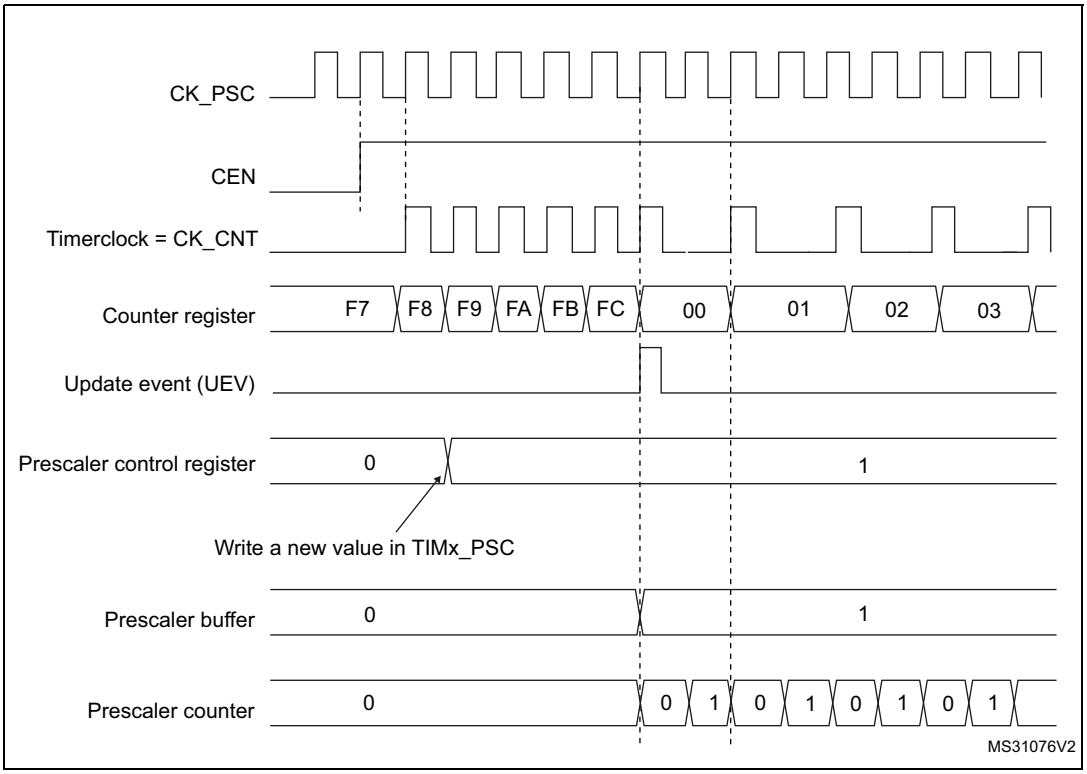
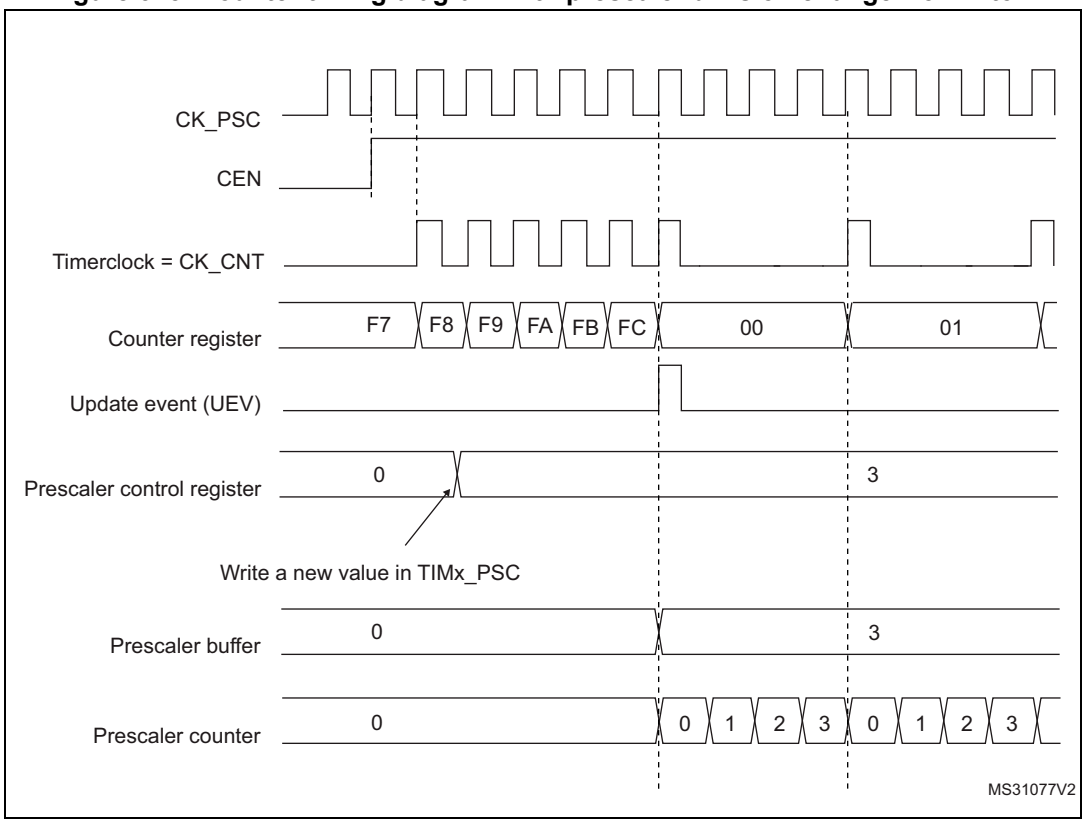


Figure 375. Counter timing diagram with prescaler division change from 1 to 4



36.3.2 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 376. Counter timing diagram, internal clock divided by 1

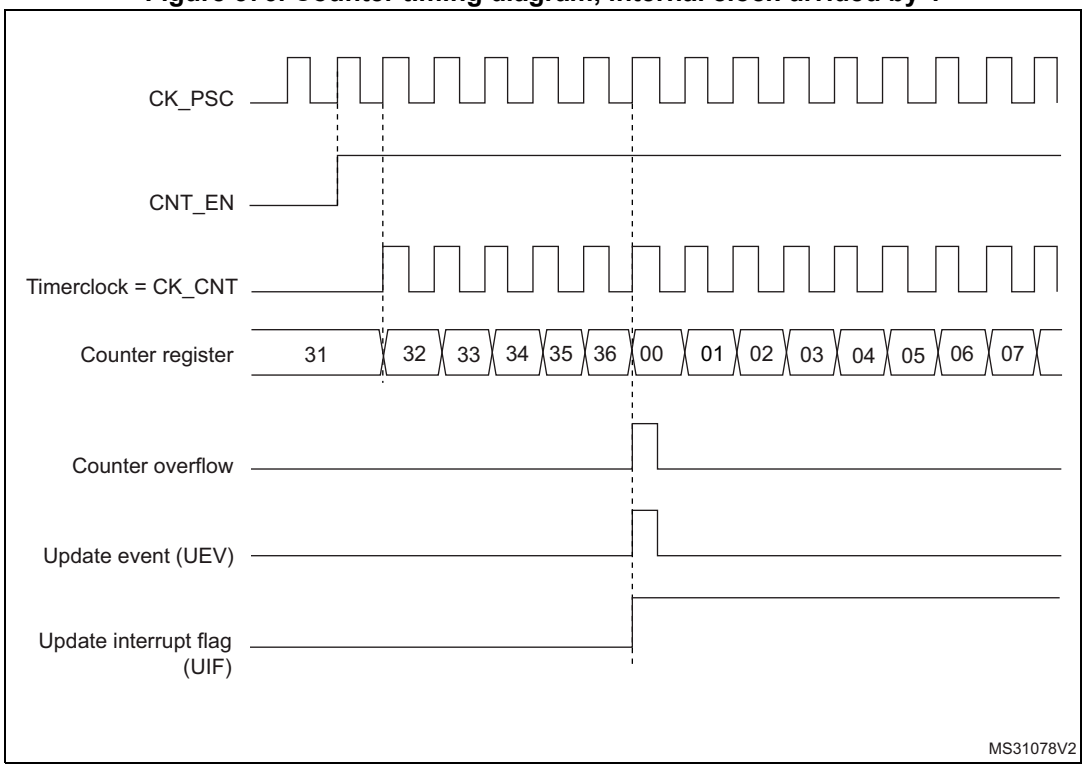


Figure 377. Counter timing diagram, internal clock divided by 2

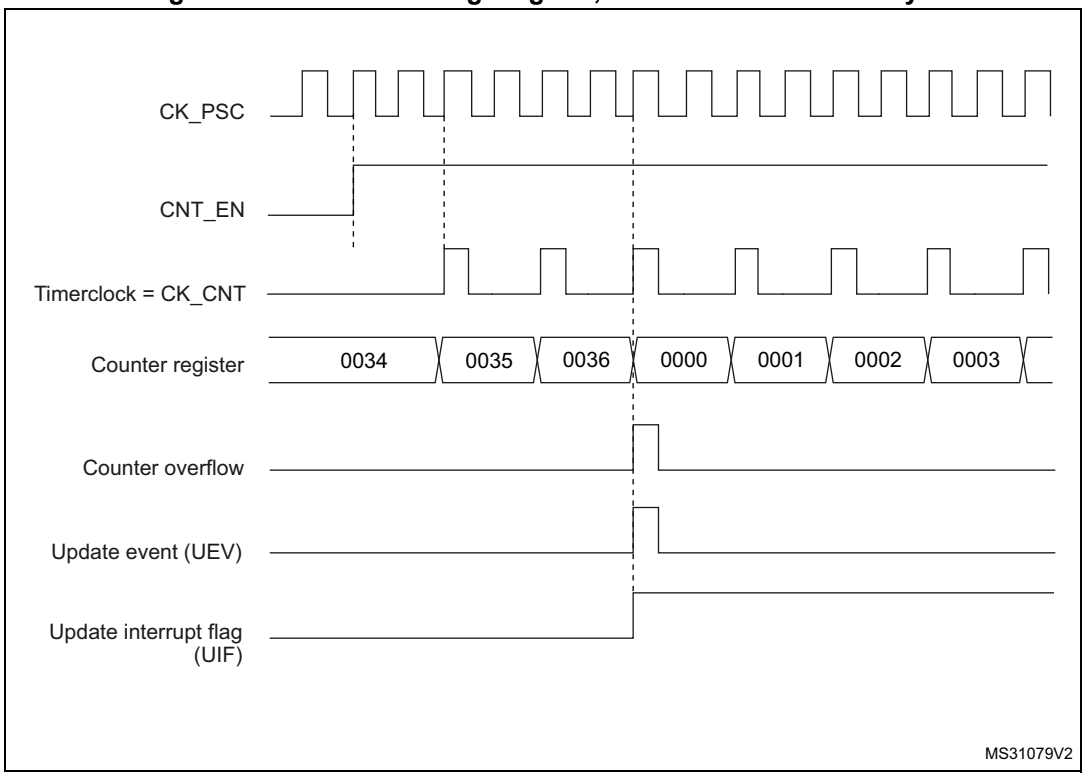


Figure 378. Counter timing diagram, internal clock divided by 4

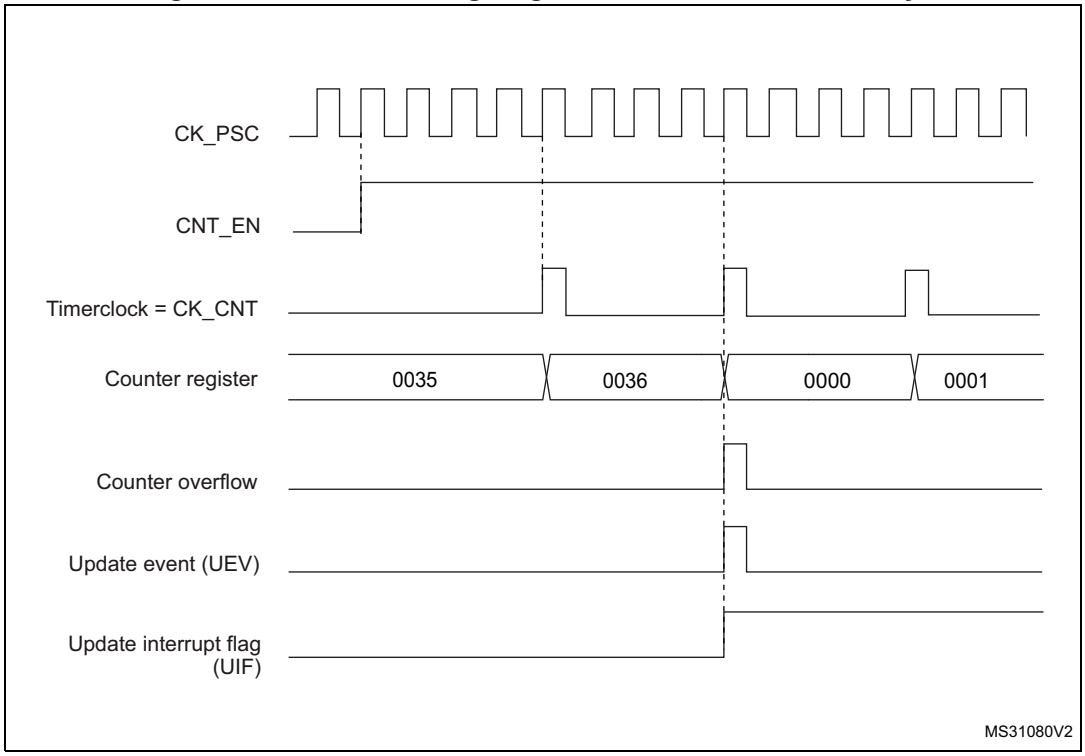


Figure 379. Counter timing diagram, internal clock divided by N

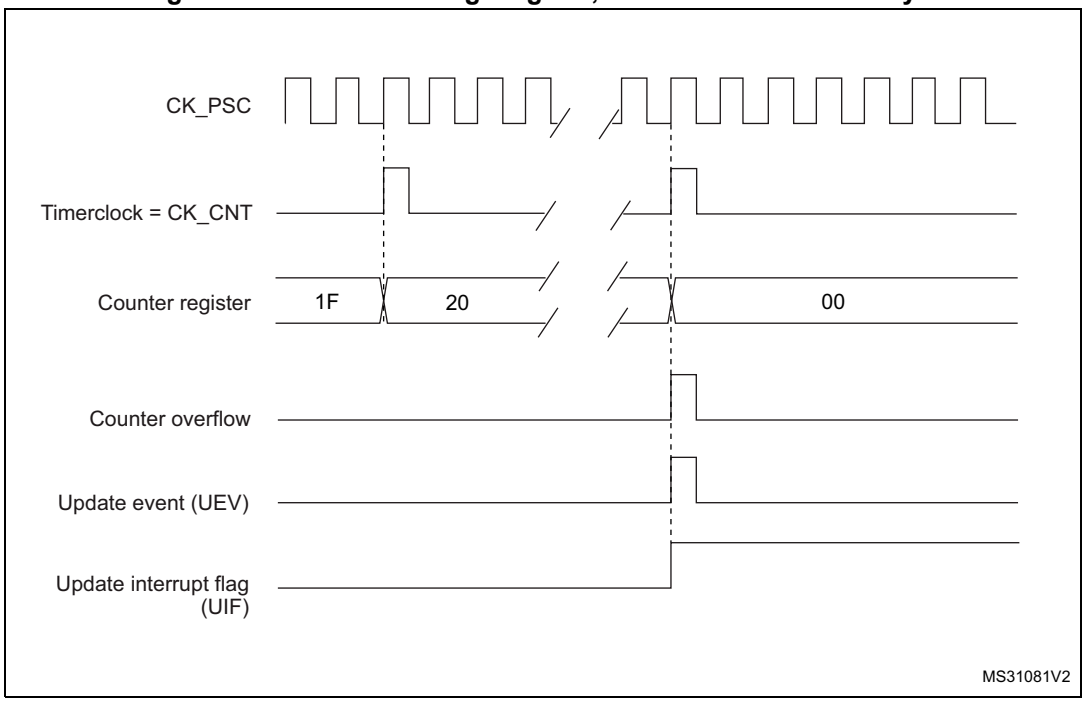


Figure 380. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

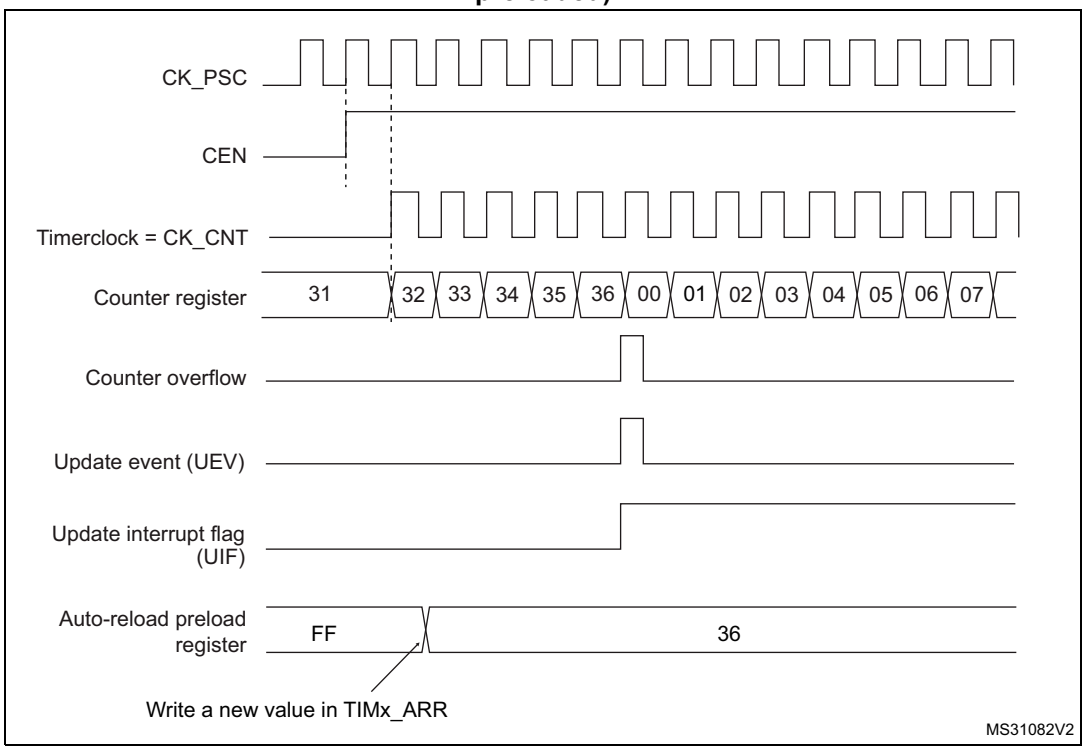
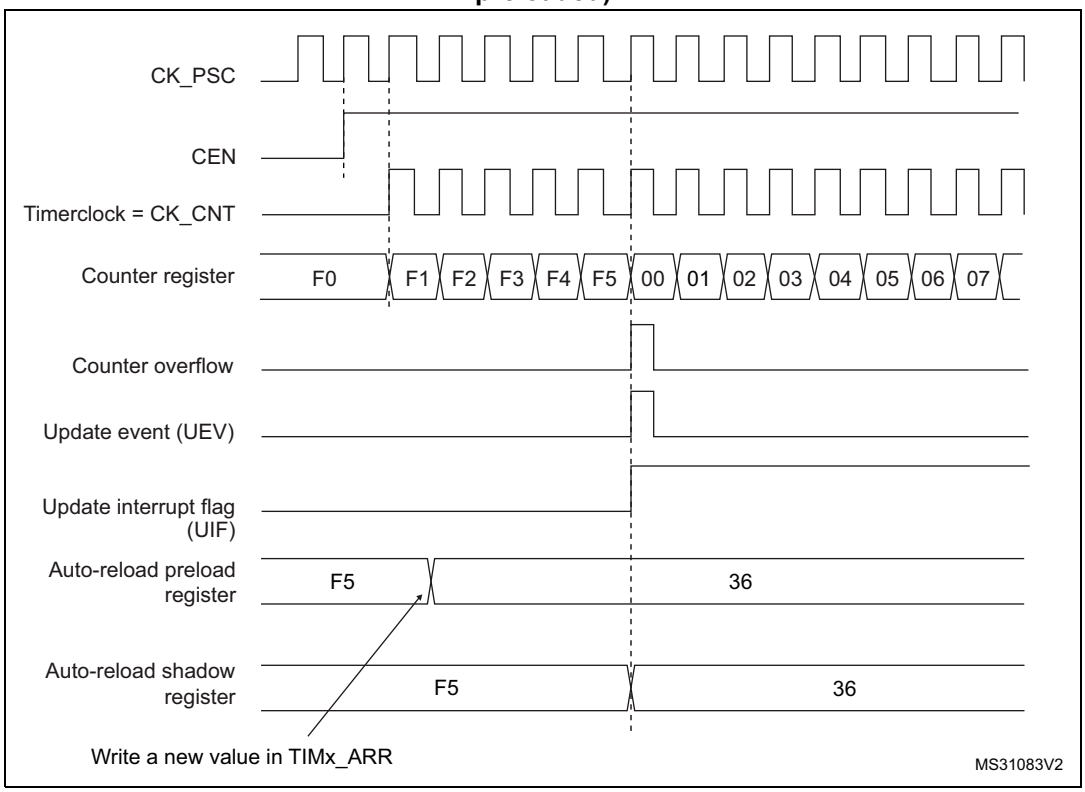


Figure 381. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



36.3.3 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.

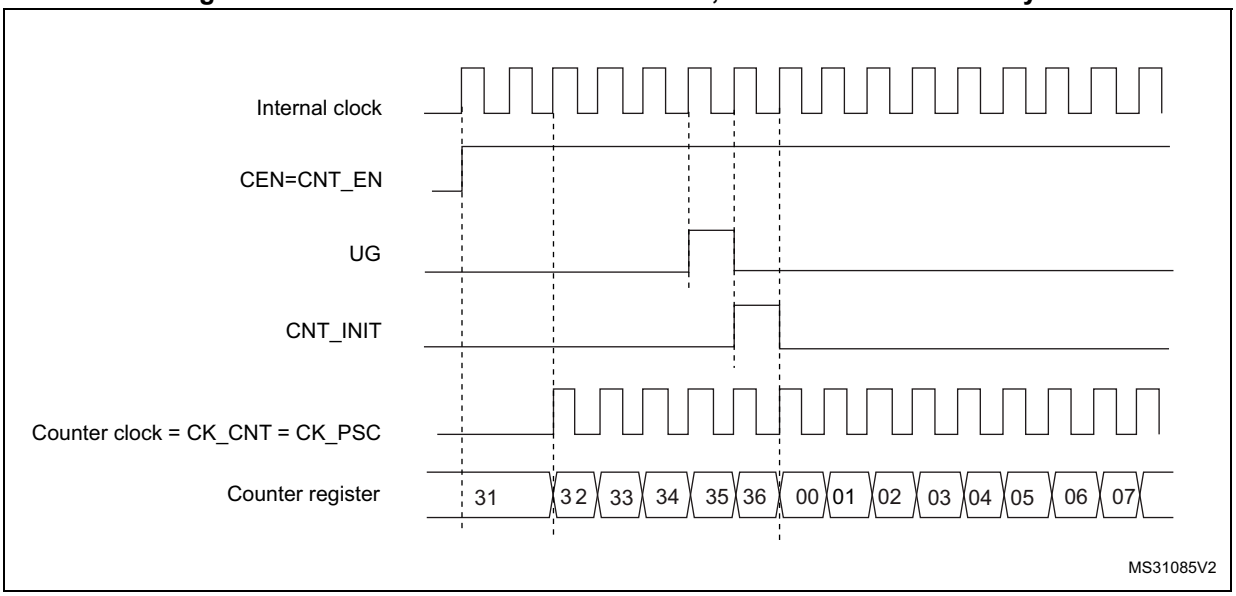
36.3.4 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

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 CK_INT.

Figure 382 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 382. Control circuit in normal mode, internal clock divided by 1



36.3.5 Debug mode

When the microcontroller enters the debug mode (Cortex®-M33 core - halted), the TIMx counter either continues to work normally or stops, depending on the DBG_TIMx_STOP configuration bit in the DBG module. For more details, refer to *Section : DBGMCU APB1 freeze register 1 (DBGMCU_APB1FZR1)*.

36.4 TIM6/TIM7 registers

Refer to *Section 1.2 on page 76* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

36.4.1 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.	Res.	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw				rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

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

Note: 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.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

36.4.2 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 (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as a trigger output (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 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.

Bits 3:0 Reserved, must be kept at reset value.

36.4.3 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.

36.4.4 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:

– At overflow or underflow regarding the repetition counter value and 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.

36.4.5 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).

36.4.6 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

36.4.7 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 CK_CNT is equal to f_{CK_PSC} / (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 or through trigger controller when configured in “reset mode”).

36.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Prescaler value
ARR is the value to be loaded into the actual auto-reload register.
Refer to *Section 36.3.1: Time-base unit on page 1344* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

36.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 284. 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.	Res.	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					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.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3 on page 87* for the register boundary addresses.

37 Low-power timer (LPTIM)

37.1 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.

37.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter

37.3 LPTIM implementation

Table 285 describes LPTIM implementation on STM32L552xx and STM32L562xx devices. The full set of features is implemented in LPTIM1. LPTIM2 and LPTIM3 support a smaller set of features, but is otherwise identical to LPTIM1.

Table 285. STM32L552xx and STM32L562xx LPTIM features

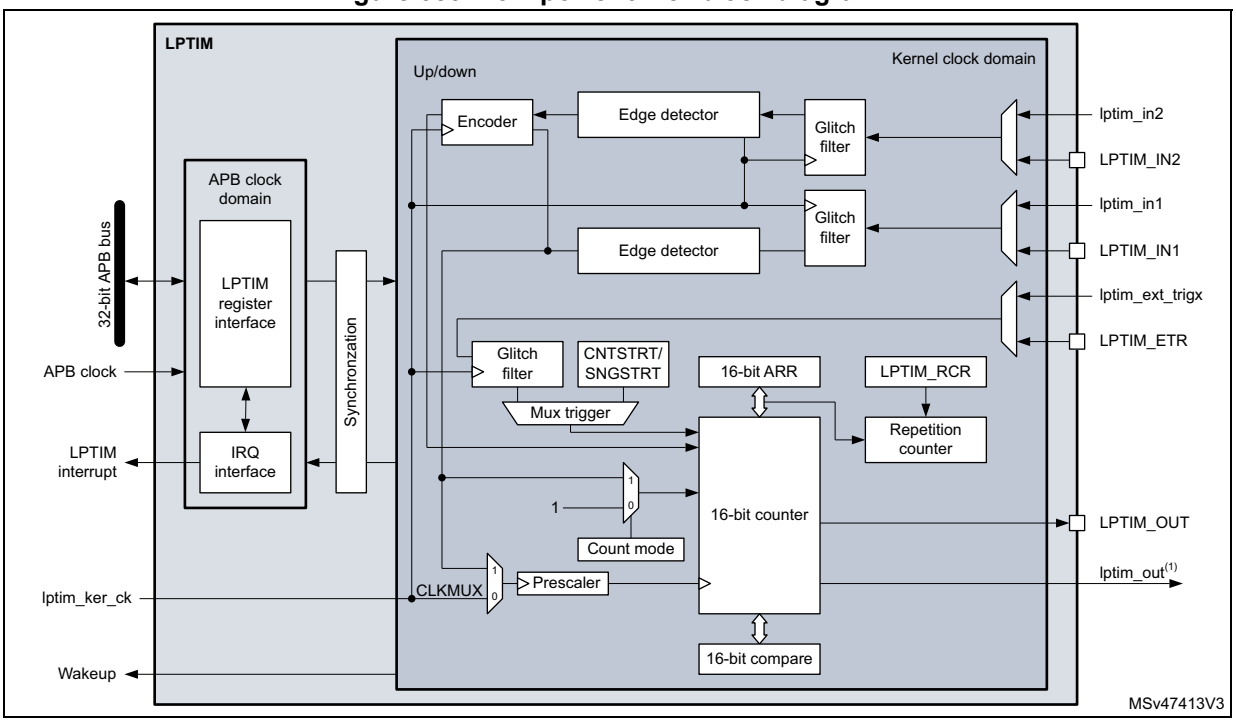
LPTIM modes/features(1)	LPTIM1	LPTIM2	LPTIM3
Encoder mode	X	-	-
External input clock	X	X	X
Wakeup from Stop	(2)	(3)	(2)

1. X = supported.
2. Wakeup supported from Stop 0, Stop 1 and Stop 2 modes.
3. Wakeup supported from Stop 0 and Stop 1 modes.

37.4 LPTIM functional description

37.4.1 LPTIM block diagram

Figure 383. Low-power timer block diagram(a)



1. lptim_out is the internal LPTIM output signal that can be connected to internal peripherals.

a. LPTIM2/LPTIM3 has only the input channel 1, no input channel 2.

37.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 286. LPTIM input/output pins

Names	Signal type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM Output GPIO pin

Table 287. LPTIM internal signals

Names	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1	Digital input	Internal LPTIM input 1
lptim_in2	Digital input	Internal LPTIM input 2(1)
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wakeup event

1. Only applies to LPTIM1

37.4.3 LPTIM trigger mapping

The LPTIM external trigger connections are detailed hereafter:

Table 288. LPTIM1 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM1_ETR alternate function
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B
lptim_ext_trig3	TAMP1 input detection
lptim_ext_trig4	TAMP2 input detection
lptim_ext_trig5	TAMP3 input detection
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

Table 289. LPTIM2 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM2_ETR alternate function
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B
lptim_ext_trig3	TAMP1 input detection
lptim_ext_trig4	TAMP2 input detection
lptim_ext_trig5	TAMP3 input detection
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

Table 290. LPTIM3 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM3_ETR alternate function
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B
lptim_ext_trig3	TAMP1 input detection
lptim_ext_trig4	TAMP2 input detection
lptim_ext_trig5	TAMP3 input detection
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

37.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal should also be provided (first configuration). In this case, the internal clock

signal frequency should be at least four times higher than the external clock signal frequency.

37.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals, such as embedded comparators), 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.

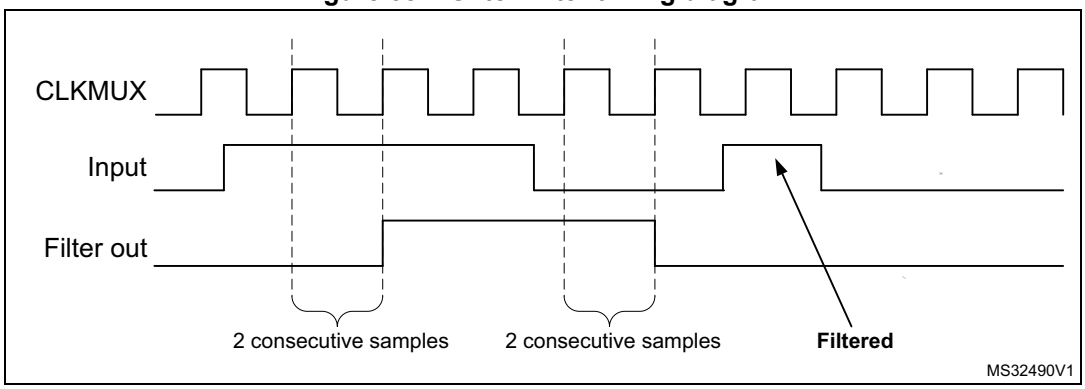
The digital filters are divided into two groups:

- 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 384* shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

Figure 384. 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.

37.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

Table 291. Prescaler division ratios

programming	dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

37.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

37.4.8 Operating mode

The LPTIM features two operating modes:

- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

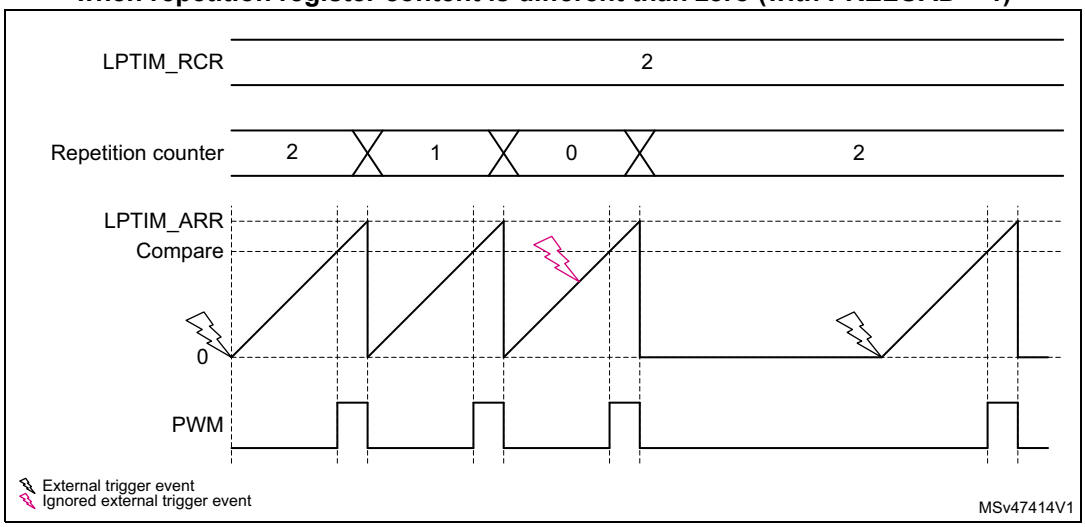
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 385*.

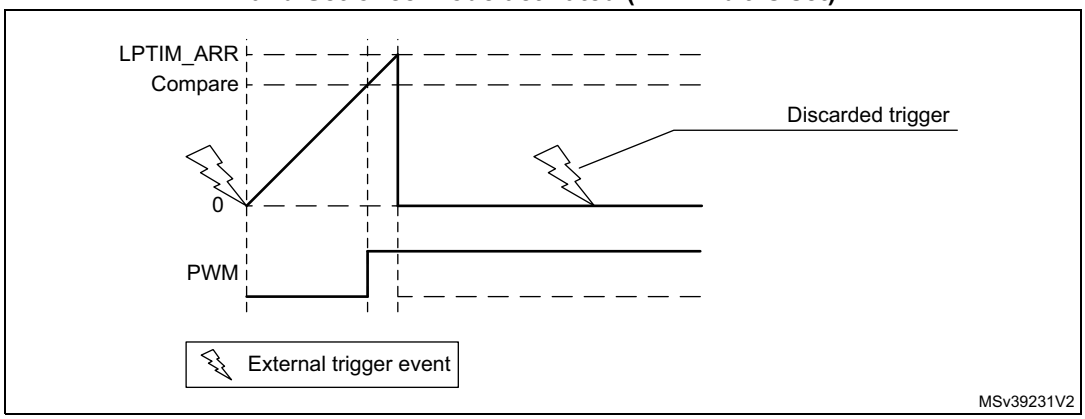
Figure 385. LPTIM output waveform, single counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

It should be noted that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 386*.

Figure 386. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = ‘00’), the SNGSTRT setting starts the counter for one-shot counting.

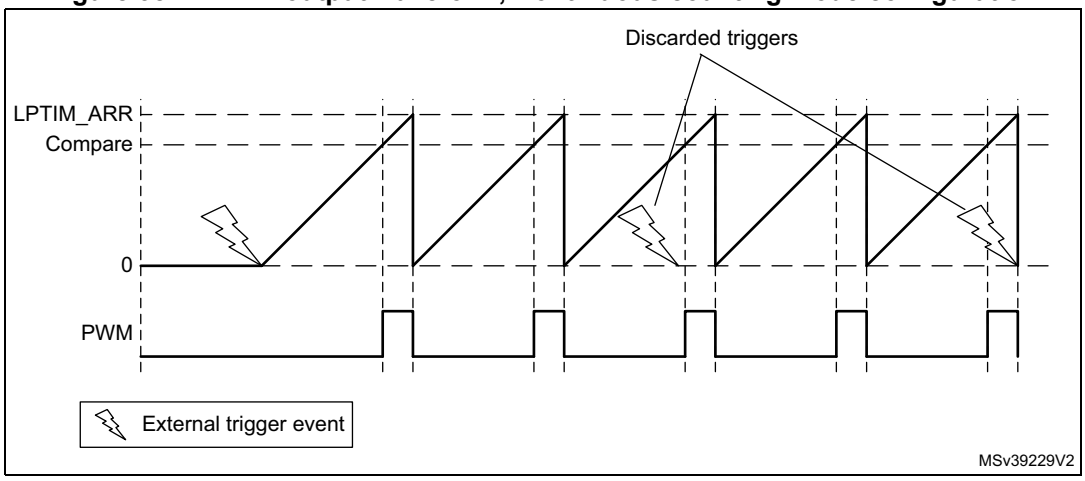
Continous mode

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, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 387*.

In case of software start (TRIGEN[1:0] = ‘00’), setting CNTSTRT starts the counter for continuous counting.

Figure 387. 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 Continous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

37.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

37.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CMP (compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CMP. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT registers.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM_ARR register value be strictly greater than the LPTIM_CMP register value.

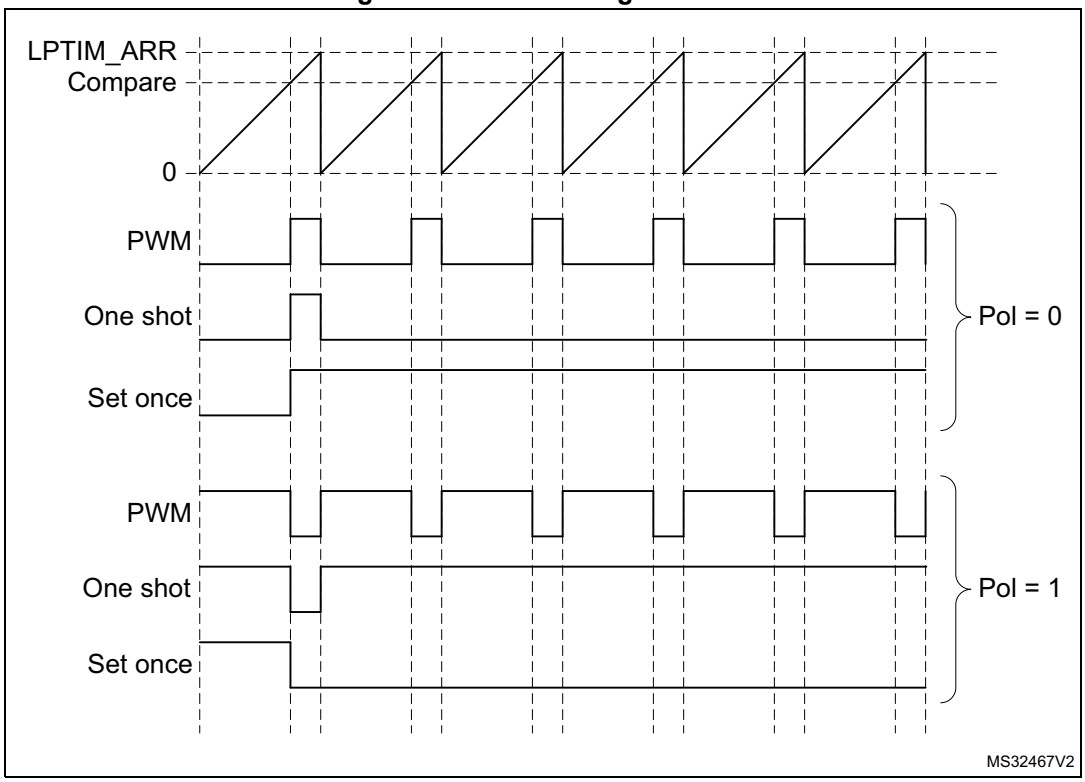
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to ‘0’ forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to ‘1’ forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. *Figure 388* 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 388. Waveform generation



37.4.11 Register update

The LPTIM_ARR register and LPTIM_CMP register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR and the LPTIM_CMP registers are updated:

- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR and the LPTIM_CMP registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR and the LPTIM_CMP registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag and the CMPOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register and the LPTIM_CMP register.

After a write to the LPTIM_ARR register or the LPTIM_CMP register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag or the CMPOK flag be set, leads to unpredictable results.

37.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

37.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM_CFGR and LPTIM_IER registers must be modified only when the LPTIM is disabled.

37.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.
- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset should be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset should be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  It should be noted that to read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So developer should make sure that these two mechanisms are used exclusively.

37.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two Down and Up flags in the LPTIM_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to ‘1’. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder’s position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

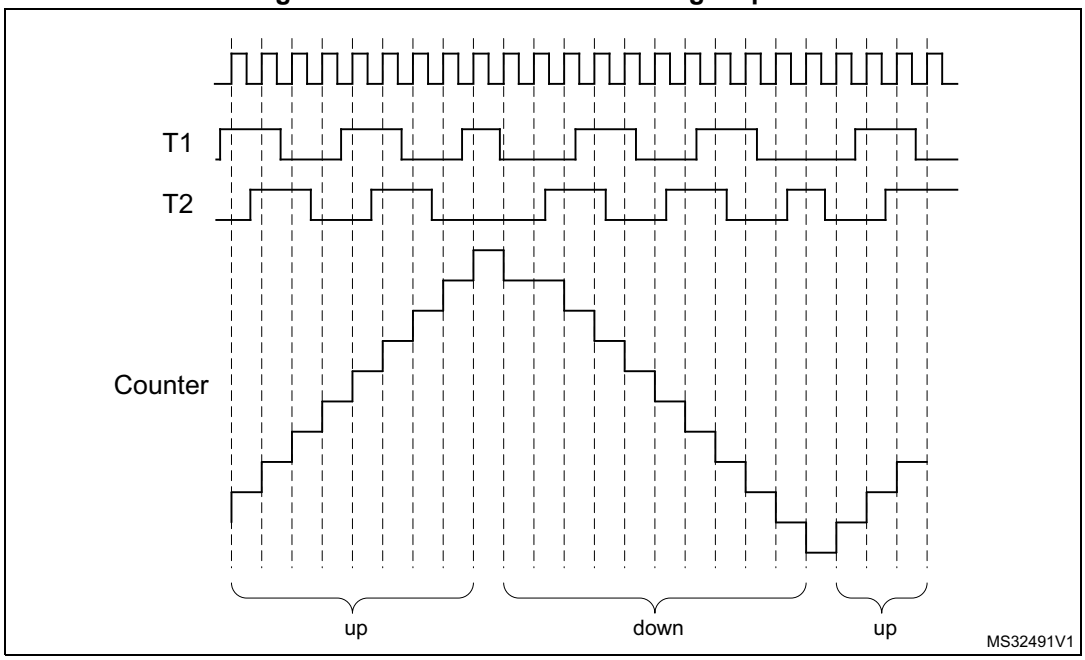
Table 292. Encoder counting scenarios

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 389. Encoder mode counting sequence



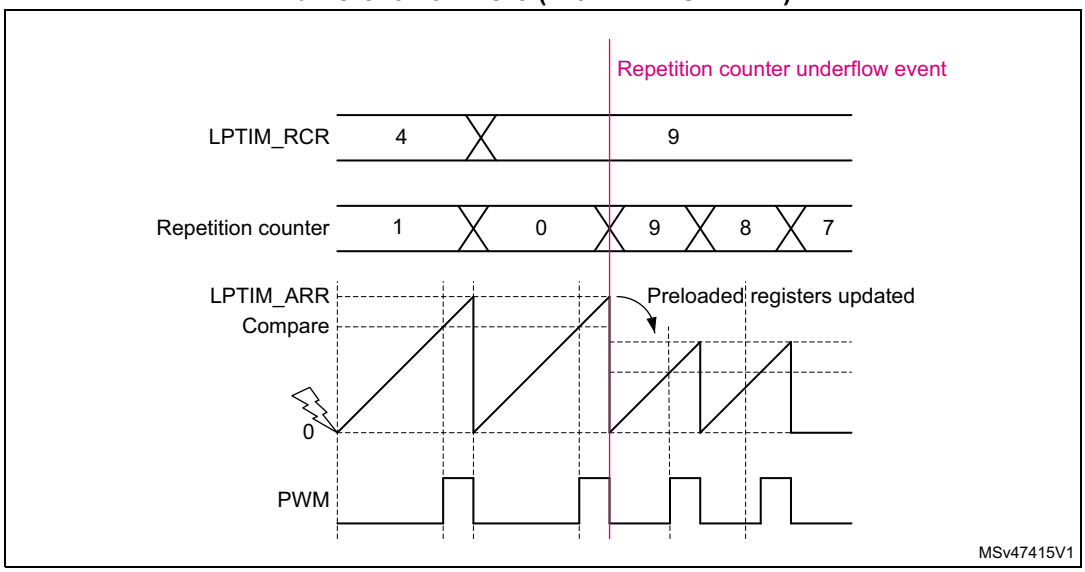
37.4.16 Repetition Counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 390*.

Figure 390. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to section "Register update" for more information).

Repetition counter underflow event is signaled to the software through the Update Event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective Update Event Interrupt Enable (UEIE) control bit, mapped to the LPTIM_IER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, that new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain so that the new written value is loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_IER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the REPOK flag is set, leads to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

37.4.17 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the DBG_LPTIM_STOP configuration bit in the DBG module.

37.5 LPTIM low-power modes

Table 293. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode (refer to *Section 37.3: LPTIM implementation*).
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

37.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_IER register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK

Note: If any bit in the LPTIM_IER register is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.

Table 294. Interrupt events

Interrupt event	Description
Compare match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the compare register (LPTIM_CMP).
Auto-reload match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the Auto-reload register (LPTIM_ARR).
External trigger event	Interrupt flag is raised when an external trigger event is detected
Auto-reload register update OK	Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete.
Compare register update OK	Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete.
Direction change	Used in Encoder mode. Two interrupt flags are embedded to signal direction change: – UP flag signals up-counting direction change – DOWN flag signals down-counting direction change.

Table 294. Interrupt events (continued)

Interrupt event	Description
Update Event	Interrupt flag is raised when the repetition counter underflows (or contains zero) and the LPTIM counter overflows.
Repetition register update Ok	REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed.

37.7 LPTIM registers

37.7.1 LPTIM interrupt and status register (LPTIM_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP OK	UE	DOWN	UP	ARR OK	CMP OK	EXT TRIG	ARRM	CMPM
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOK**: Repetition register update Ok

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 37.3: 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 37.3: 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.

37.7.2 LPTIM interrupt clear register (LPTIM_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
							w	w	w	w	w	w	w	w	w

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 37.3: 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 37.3: 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 CMP flag in the LPTIM_ISR register

37.7.3 LPTIM interrupt enable register (LPTIM_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 37.3: 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 37.3: 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')

37.7.4 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	WAV POL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable

The ENC bit controls the Encoder mode
- 0: Encoder mode disabled
- 1: Encoder mode enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 37.3: 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, LPTIM_RCR and the LPTIM_CMP registers update modality
- 0: Registers are updated after each APB bus write access
- 1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity

The WAVEPOL bit controls the output polarity

0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers

1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers

Bit 20 **WAVE**: Waveform shape

The WAVE bit controls the output shape

0: Deactivate Set-once mode

1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable

The TIMOUT bit controls the Timeout feature

0: A trigger event arriving when the timer is already started is ignored

1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity

The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:

00: software trigger (counting start is initiated by software)

01: rising edge is the active edge

10: falling edge is the active edge

11: both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0

001: lptim_ext_trig1

010: lptim_ext_trig2

011: lptim_ext_trig3

100: lptim_ext_trig4

101: lptim_ext_trig5

110: lptim_ext_trig6

111: lptim_ext_trig7

See *Section 37.4.3: LPTIM trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1

001: /2

010: /4

011: /8

100: /16

101: /32

110: /64

111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any trigger active level change is considered as a valid trigger

01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.

10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.

11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any external clock signal level change is considered as a valid transition

01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock Polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: the rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 1 is active.

01: the falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 2 is active.

10: both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 3 is active.

11: not allowed

Refer to *Section 37.4.15: Encoder mode* for more details about Encoder mode sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

37.7.5 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software should consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in Continuous mode.
If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.
If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in single pulse mode.
If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.
If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.
This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.
0: LPTIM is disabled.
1: LPTIM is enabled

37.7.6 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’).

37.7.7 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’).

37.7.8 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.

37.7.9 LPTIM1 option register (LPTIM1_OR)

Address offset: 0x020

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OR_1**: Option register bit 1

0: LPTIM1 input 2 is connected to I/O

1: LPTIM1 input 2 is connected to COMP2_OUT

Bit 0 **OR_0**: Option register bit 0

0: LPTIM1 input 1 is connected to I/O

1: LPTIM1 input 1 is connected to COMP1_OUT

37.7.10 LPTIM2 option register (LPTIM2_OR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **OR_[1:0]**

00: input 1 is connected to I/O
01: input 1 is connected to COMP1_OUT
10: input 1 is connected to COMP2_OUT
11: input 1 is connected to COMP1_OUT OR COMP2_OUT

37.7.11 LPTIM3 option register (LPTIM3_OR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **OR_[1:0]**

00: input 1 is connected to I/O
01: input 1 is connected to COMP1_OUT
10: input 1 is connected to COMP2_OUT
11: input 1 is connected to COMP1_OUT OR COMP2_OUT

37.7.12 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

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

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

Bits 7:0 **REP[7:0]**: Repetition register value
REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

Reset value: 0x0000 0020

37.7.13 LPTIM register map

The following table summarizes the LPTIM registers.

Table 295. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN(1)	UP(1)	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
	Reset value																								0	0	0	0	0	0	0	0	0
0x004	LPTIM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF(1)	UPCF(1)	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
	Reset value																								0	0	0	0	0	0	0	0	0
0x008	LPTIM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE(1)	UPIE(1)	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
	Reset value																								0	0	0	0	0	0	0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(1)	COUNTMODE	PRELOAD	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
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	LPTIM1_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
	Reset value																															0	0
0x020	LPTIM2_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
	Reset value																															0	0

Table 295. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	LPTIM3_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_1	OR_0
	Reset value																															0	0
0x028	LPTIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

1. *If LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 37.3: LPTIM implementation.*

Refer to *Section 2.3 on page 87* for the register boundary addresses.

38 Infrared interface (IRTIM)

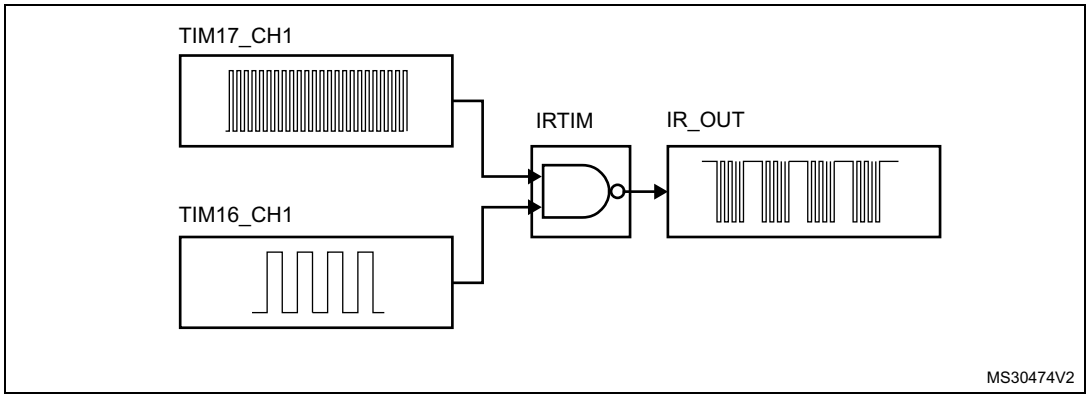
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16 and TIM17 as shown in *Figure 391*.

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 391. IRTIM internal hardware connections with TIM16 and TIM17



All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 generates the modulation envelope.

The infrared function is output on the IR_OUT pin. The activation of this function is done through the GPIOx_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 pin) can be activated through the I2C_PB9_FMP bit in the SYSCFG_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

39 Independent watchdog (IWDG)

39.1 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 40 on page 1395*.

39.2 IWDG main features

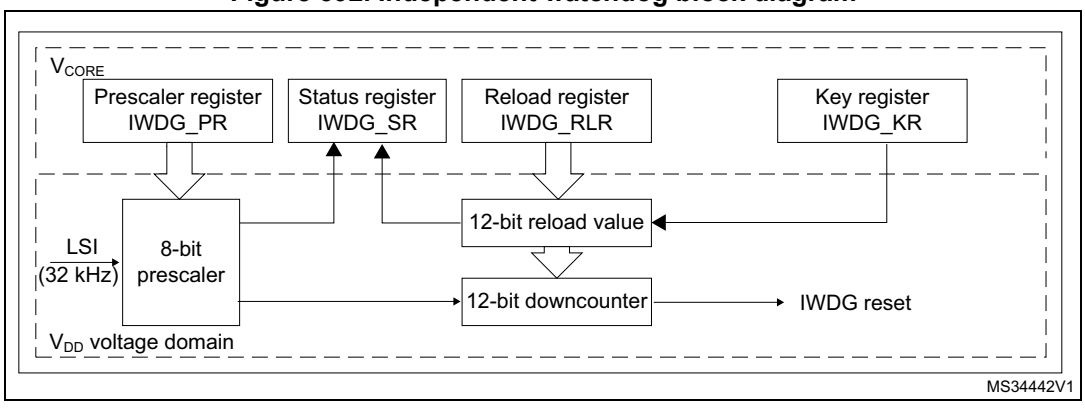
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

39.3 IWDG functional description

39.3.1 IWDG block diagram

Figure 392 shows the functional blocks of the independent watchdog module.

Figure 392. Independent watchdog block diagram



1. The register interface is located in the V_{CORE} voltage domain. The watchdog function is located in the V_{DD} voltage domain, still functional in Stop and Standby modes.

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.

39.3.2 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).

39.3.3 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.

39.3.4 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 *Section 6.4: Flash memory option bytes* for more details.

39.3.5 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.

39.3.6 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.

39.4 IWDG registers

Refer to *Section 1.2 on page 76* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

39.4.1 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 39.3.5: Register access protection*)

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

39.4.2 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 39.3.5: 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.

39.4.3 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.

39.4.4 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 LSI/Prescaler clock 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 LSI/Prescale clock 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 LSI/Prescale clock 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.

39.4.5 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 39.3.5*, 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.

39.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

Table 296. 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 2.3 on page 87* for the register boundary addresses.

40 System window watchdog (WWDG)

40.1 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.

40.2 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 394*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40.

40.3 WWDG functional description

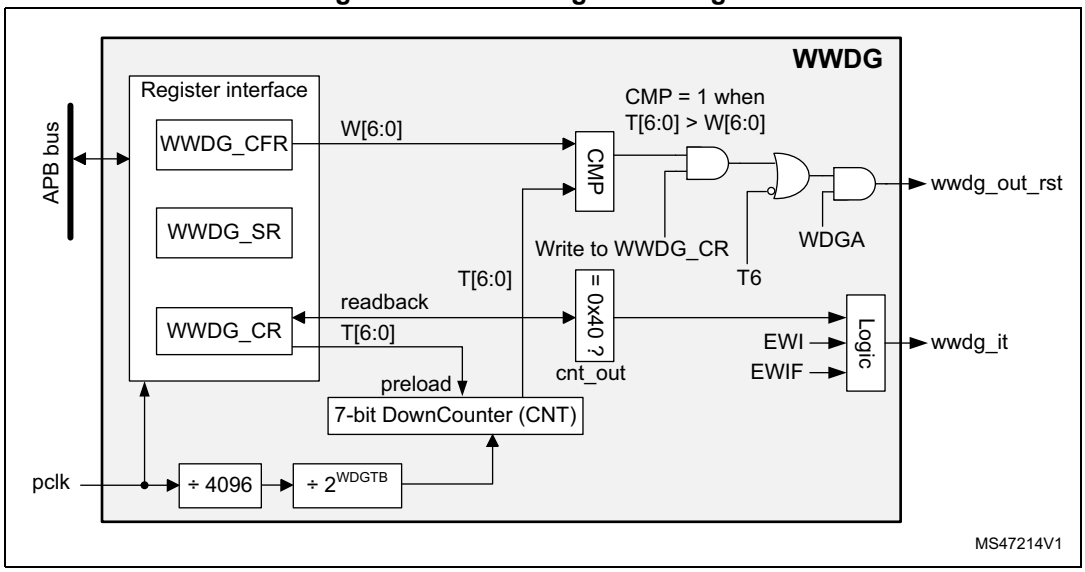
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, 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 393* for the WWDG block diagram.

40.3.1 WWDG block diagram

Figure 393. Watchdog block diagram



40.3.2 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

40.3.3 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 394*). 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 394* describes the window watchdog process.

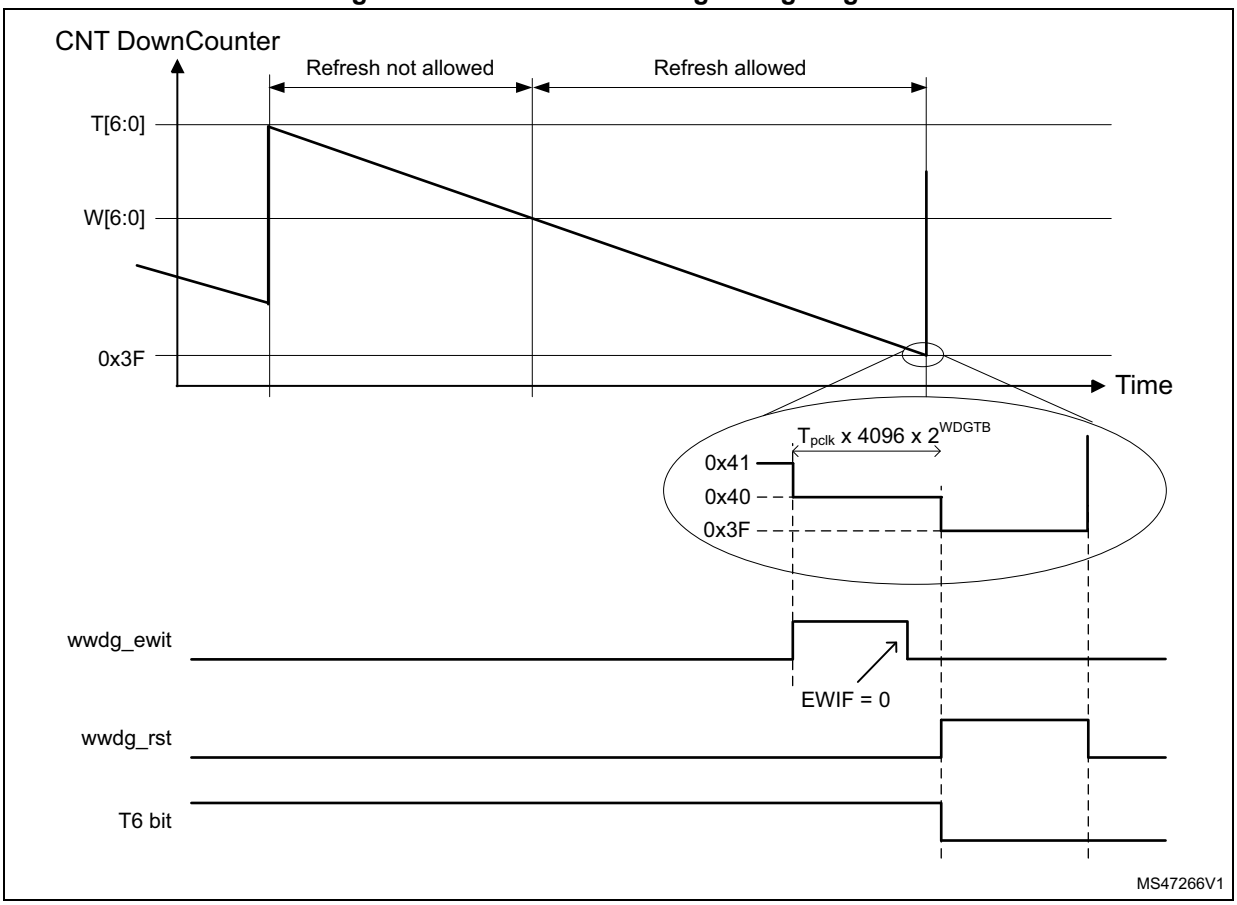
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

40.3.4 How to program the watchdog timeout

Use the formula in *Figure 394* 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 394. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[1:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

t_{WWDG}: WWDG timeout

t_{PCLK}: APB clock period measured in ms

4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[1:0] 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}.

40.3.5 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 : DBGMCU APB1 freeze register 1 (DBGMCU_APB1FZR1)*.

40.4 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.

40.5 WWDG registers

Refer to *Section 1.2 on page 76* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

40.5.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

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

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

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[1:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

40.5.2 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.	Res.	Res.	Res.	Res.	EWI	WDGTB[1:0]		W[6:0]						
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 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 **WDGTB[1:0]:** Timer base

The time base of the prescaler can be modified as follows:
00: CK counter clock (PCLK div 4096) div 1
01: CK counter clock (PCLK div 4096) div 2
10: CK counter clock (PCLK div 4096) div 4
11: CK counter clock (PCLK div 4096) div 8

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

40.5.3 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.

40.5.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 297. 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.	Res.	Res.	Res.	Res.	EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							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 2.3 on page 87* for the register boundary addresses.

41 Real-time clock (RTC)

41.1 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.

41.2 RTC main features

The RTC supports the following features (see *Figure 395: 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.
- TrustZone support:
  - RTC fully securable
  - Alarm A, alarm B, wakeup Timer and timestamp individual secure or non-secure configuration

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.

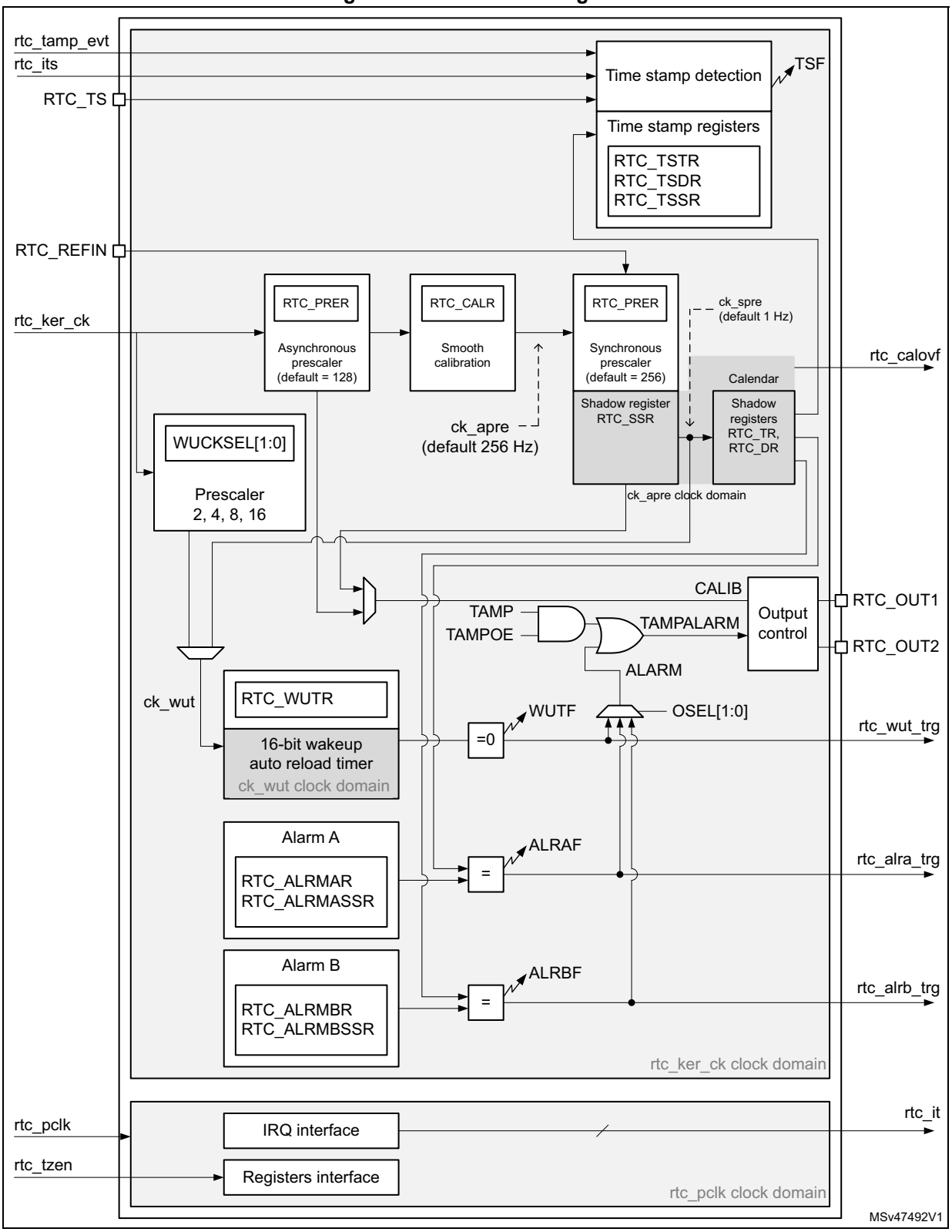
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE. When clocked by the LSI, the RTC is not functional in V_{BAT} mode, but is functional in all low-power modes except Shutdown mode.

All RTC events (Alarm, WakeUp Timer, Timestamp) can generate an interrupt and wakeup the device from the low-power modes.

41.3 RTC functional description

41.3.1 RTC block diagram

Figure 395. RTC block diagram



41.3.2 RTC pins and internal signals

Table 298. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 select one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between TAMP and ALARM outputs.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wakeup outputs. TAMP is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 299. 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_tzen	Input	RTC TrustZone enabled
rtc_it	Output	RTC interrupts (refer to *Section 41.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wakeup timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or V_{BAT} when the selected clock is not LSE. Refer to *Section 41.4: RTC low-power modes* for more details.

Table 300. 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_tzen	From FLASH option bytes: TZEN
rtc_wut_trg	To temperature sensor power-on (when cleared) and down (when set)
rtc_calovf	To TAMP peripheral: tamp_itamp5

The TZEN option bit is used to activate TrustZone in the device.

TZEN = 1: TrustZone activated.

TZEN = 0: TrustZone disabled.

When TrustZone is disabled, the APB access to the RTC registers are non-secure.

The triggers outputs can be used as triggers for other peripherals.

41.3.3 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 42: Tamper and backup registers (TAMP)*).

RTC_OUT1, RTC_TS, TAMP_IN1 and TAMP_OUT2 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 301*.

Table 301. RTC pin 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	TAMP2E & TAMP2AM	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	Don't care
	00	1							
	01 or 10 or 11	1							

Table 301. RTC pin 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	TAMP2E & TAMP2AM	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	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	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	Don't care
TAMP_OUT2 output PP		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care
TAMP_IN1 input floating		00	0	0	Don't care	Don't care	Don't care	0	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	0	1	1
		00	0	1	1			0		
		Don't care	Don't care	0				0		
RTC_TS input floating		00	0	0	Don't care	Don't care	Don't care	0	0	1
		00	0	1	1			0		
		Don't care	Don't care	0				0		

Table 301. RTC pin 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	TAMP2E & TAMP2AM	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	0
	00	0	1	1			0		
	Don't care	Don't care	0				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 shown in table *Table 302*.

Table 302. 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

41.3.4 RTC secure protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both secure and non-secure modes, except for the RTC secure mode control register (RTC_SMCR) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the DECPROT bit is cleared in the RTC_SMCR register:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SMCR, RTC_PRIVCR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in secure and non-secure modes. All the other RTC registers can be read only in secure mode.

When the DECPROT bit is set, it is still possible to protect some of the registers by clearing dedicated INITDPROT, CALDPROT, TSDPROT, WUTDPROT, ALRADPROT or ALRBDPROT control bits. If all these bits are also set, all the RTC registers can be read and written in secure and non-secure mode.

- When INITDPROT is cleared:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT in RTC_ICSR, FMT control bits in RTC_CR and INITPRIV in the RTC_PRIVCR can be written only in secure mode.
  - These registers and control bits can be read in secure and non-secure mode.
- When CALDPROT is cleared:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR and CALPRIV in the RTC_PRIVCR can be written only in secure mode.
  - These registers and control bits can be read in secure and non-secure mode.
- When ALRADPROT is cleared:
  - RTC_ALRMAR, RTC_ALRMASSR registers, plus ALRAE, ALRAIE in the RTC_CR, CALRAF in the RTC_SCR, ALRAF in RTC_SR, and ALRAMF in RTC_SMISR can be read and written only in secure mode.
  - ALRAPRIV in the RTC_PRIVCR can be written only in secure mode
- When ALRBDPROT is cleared:
  - RTC_ALRMBR, RTC_ALRMBSSR registers, plus ALRBE and ALRBIE in the RTC_CR, CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_SMISR can be read and written only in secure mode.
  - ALRBPRIV in the RTC_PRIVCR can be written only in secure mode.
- When WUTDPROT is cleared:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_SMISR can be read and written only in secure mode.
  - WUTPRIV in the RTC_PRIVCR can be written only in secure mode
- When TSDPROT is cleared:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_SMISR can be read and written only in secure mode.
  - TSPRIV in the RTC_PRIVCR can be written only in secure mode

A non-secure access to a secure-protected register is denied:

- There is no bus error generated.
- In case the register has a global protection: a notification is generated through a flag/interrupt in the TZIC (TrustZone illegal access controller). In case only a few bits of

the register are protected (for registers with mixed features such as RTC_CR...), no notification is generated.

- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the RTC reset and clock control is also secured in the RCC.

41.3.5 RTC privilege protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both privileged and non-privileged modes, except for the RTC privilege mode control register (RTC_PRIVCR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_SMCR, RTC_PRIVCR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and non-privilege modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and non-privilege modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT in RTC_ICSR and FMT control bits in RTC_CR, plus INITDPROT in the RTC_SMCR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR, plus CALDPROT in the RTC_SMCR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privilege mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE and ALRAIE in the RTC_CR, and CALRAF in the RTC_SCR, ALRAF in RTC_SR, and ALRAMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRADPROT in the RTC_SMCR can be written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE and ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in RTC_SR,

and ALRBMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRBDPROT in the RTC_SMCR can be written only in privilege mode.
- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - WUTDPROT in the RTC_SMCR can be written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - TSDPROT in the RTC_SMCR can be written only in privilege mode.

A non-privileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

41.3.6 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to “Reset and clock control (RCC)”.

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 395: 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 41.3.10: Periodic auto-wakeup* for details).

41.3.7 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 41.6.12: 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.

41.3.8 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre instead of RTCCLK or ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : Calibration ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

41.3.9 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.

41.3.10 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this allows the wakeup interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock).

  When ck_spre frequency is 1 Hz, this allows a wakeup time to be achieved from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wakeup timer on page 1415*), the timer starts counting down. When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000<WUTOCLR<=WUT).

The wakeup flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 41.3.1: RTC block diagram*).

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

41.3.11 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: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bit INIT in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

The registers protected by INITPRIV are write-protected by the INIT KEY.

The registers protected by CALDPRIV are write-protected by the CAL KEY.

In case PRIV or INITPRIV is set in the RTC_PRIVCR, and/or DPROT or INITDPROT is cleared in the RTC_SMCR: the INIT KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege and security mode defined by PRIV, INITPRIV, DPROT, INITDPROT configuration.

In case PRIV or CALPRIV is set in the RTC_PRIVCR, and/or DPROT or CALDPROT is cleared in the RTC_SMCR: the CAL KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege and security mode defined by PRIV, CALPRIV, DPROT, CALDPROT configuration.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.

If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.

3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting.

RTC_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode.
   If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wakeup auto-reload value WUT[15:0], WUTOCLR[15:0] and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting.
   If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

41.3.12 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 1413*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 41.3.14: 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.

41.3.13 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.

41.3.14 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock.

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: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow occurs.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

41.3.15 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.

41.3.16 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

Calibration ultra-low-power mode

The calibration consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL=0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are

set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL=0, in order to have a 32 second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

41.3.17 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).

41.3.18 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. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

41.3.19 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set is the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wakeup flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

When the TAMPALRM output is selected, the RTC_OUT1 pin is automatically configured but the RTC_OUT2 pin must be set as alternate function. In case the TAMPALRM is configured open-drain in the RTC, the RTC_OUT1 GPIO must be configured as input.

41.4 RTC low-power modes

Table 303. 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 304. 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	Yes	No

41.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration. The interrupt output or the secure interrupt output is also activated.

Table 305. Non-secure 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 and (ALRADPROT=1 and DECPROT=1)	write 1 in CALRAF	Yes	Yes(3)	Yes(4)
	Alarm B	ALRBF	ALRBIE and (ALRBDPROT=1 and DECPROT=1)	write 1 in CALRBF	Yes	Yes(3)	Yes(4)
	Timestamp	TSF	TSIE and (TSDPROT=1 and DECPROT=1)	write 1 in CTSF	Yes	Yes(3)	Yes(4)
	Wakeup timer	WUTF	WUTIE and (WUTDPROT=1 and DECPROT=1)	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.

Table 306. Secure 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_S	Alarm A	ALRAF	ALRAIE and (ALRADPROT=0 or DECPROT=0)	write 1 in CALRAF	Yes	Yes(3)	Yes(4)
	Alarm B	ALRBF	ALRBIE and (ALRBDPROT=0 or DECPROT=0)	write 1 in CALRBF	Yes	Yes(3)	Yes(4)
	Timestamp	TSF	TSIE and (TSDPROT=0 or DECPROT=0)	write 1 in CTSF	Yes	Yes(3)	Yes(4)
	Wakeup timer	WUTF	WUTIE and (WUTDPROT=0 or DECPROT=0)	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.

41.6 RTC registers

Refer to *Section 1.2 on page 76* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

41.6.1 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 1413* and *Reading the calendar on page 1415*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1413*.

This register can be write-protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: 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

41.6.2 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 1413* and *Reading the calendar on page 1415*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1413*.

This register can be write-protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset: 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.

41.6.3 RTC sub second register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: 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.

41.6.4 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 1413*.

This register can be globally protected, or each bit of this register can be individually protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
								rw	r	rc_w0	r	r	r		

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

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:8 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_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Wakeup timer configuration update not allowed except in initialization mode

1: Wakeup timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

41.6.5 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 1413*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1413*.

This register can be write-protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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)

41.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

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

Bits 31:16 **WUTOCLR[15:0]**: Wakeup auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

41.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1413.

This register can be globally protected, or each bit of this register can be individually protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	Res.	Res.	TAMPOE	TAMPTS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPSHAD	REFCKON	TSEDGE	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 the RTC outputs to be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bits 28:27 Reserved, must be kept at reset value.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM

0: The tamper flag is not routed on TAMPALRM
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a RTC timestamp to be saved
1: Save RTC timestamp on tamper detection event
TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set after the tamper flags, therefore if TAMPTS and TSIE are set, it is recommended to disable the tamper interrupts in order to avoid servicing 2 interrupts.

Bit 24 **ITSE**: timestamp on internal event enable
0: internal event timestamp disabled
1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable
This bit enables the CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection
These bits are used to select the flag to be routed to TAMPALRM output.
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wakeup output enabled

Bit 20 **POL**: Output polarity
This bit is used to configure the polarity of TAMPALRM output.
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection
When COE = 1, this bit selects which signal is output on CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 41.3.18: Calibration clock output*.

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable
0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable
0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable
0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wakeup timer enable
0: Wakeup timer disabled
1: Wakeup timer enabled

Bit 9 **ALRBE**: Alarm B enable
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 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. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

41.6.8 RTC privilege mode control register (RTC_PRIVCR)

This register can be written only when the APB access is privileged. This register can be write-protected, or each bit of this register can be individually write-protected against non-secure access depending on the RTC_SMCR configuration (refer to *Section 41.3.5: RTC privilege protection modes*).

Address offset: 0x1C

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
PRIV	INIT PRIV	CAL PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS PRIV	WUT PRIV	ALRB PRIV	ALRA PRIV
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIV**: RTC privilege protection

0: All RTC registers can be written when the APB access is privileged or non-privileged, except the registers protected by other privilege protection bits.

1: All RTC registers can be written only when the APB access is privileged.

Bit 14 **INITPRIV**: Initialization privilege protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or non-privileged.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, Delight saving, calibration and reference clock privilege protection

0: Shift register, Delight saving, calibration and reference clock can be written when the APB access is privileged or non-privileged.

1: Shift register, Delight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSPRIV**: Timestamp privilege protection

0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or non-privileged.

1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 **WUTPRIV**: Wakeup timer privilege protection

0: RTC Wakeup timer configuration and interrupt clear can be written when the APB access is privileged or non-privileged.

1: RTC Wakeup timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 **ALRBPRIV**: Alarm B privilege protection

0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or non-privileged.

1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 **ALRAPRIV**: Alarm A privilege protection

0: RTC Alarm A configuration and interrupt clear can be written when the APB access is privileged or non-privileged.

1: RTC Alarm A configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 41.3.5: RTC privilege protection modes for details on the read protection.

41.6.9 RTC secure mode control register (RTC_SMCR)

This register can be written only when the APB access is secure.

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the RTC_PRIVCR configuration (refer to *Section 41.3.5: RTC privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x0000 E00F

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
DEC PROT	INIT DPROT	CAL DPROT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS DPROT	WUT DPROT	ALRB DPROT	ALRA DPROT
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DECPROT**: RTC global protection
0: All RTC registers can be written only when the APB access is secure.
1: All RTC registers can be written when the APB access is secure or non-secure, except the registers protected by other secure protection bits.

Bit 14 **INITDPROT**: Initialization protection
0: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.
1: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or non-secure.

Bit 13 **CALDPROT**: Shift register, daylight saving, calibration and reference clock protection
0: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.
1: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or non-secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSDPROT**: Timestamp protection
0: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.
1: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or non-secure.

Bit 2 **WUTDPROT**: Wakeup timer protection
0: RTC wakeup timer configuration and interrupt clear can be written only when the APB access is secure.
1: RTC wakeup timer configuration and interrupt clear can be written when the APB access is secure or non-secure.

Bit 1 **ALRBDPROT**: Alarm B protection
0: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.
1: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or non-secure.

Bit 0 **ALRADPROT**: Alarm A protection
0: RTC alarm A configuration and interrupt clear can be written only when the APB access is secure.
1: RTC alarm A configuration and interrupt clear can be written when the APB access is secure or non-secure.

Note: Refer to Section 41.3.4: RTC secure protection modes for details on the read protection.

41.6.10 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.

41.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1413*.

This register can be write-protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 × CALP) - CALM.

Refer to *Section 41.3.16: 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 41.3.16: 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 41.3.16: RTC smooth digital calibration.

Bit 12 **LPCAL**: Calibration low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode should be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 41.3.16: RTC smooth digital calibration on page 1418*.

41.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1413*.

This register can be protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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.

41.6.13 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.

This register can be protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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.

41.6.14 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.

This register can be protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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

41.6.15 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.

This register can be protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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.

41.6.16 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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.

41.6.17 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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.

41.6.18 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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

41.6.19 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 can be protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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.

41.6.20 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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.
If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wakeup auto-reload counter reaches WUTOCLR value.
If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag
This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag
This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared 2 APB clock cycles after setting their corresponding clear bit in the RTC_SCR register.

41.6.21 RTC non-secure masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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 non-secure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp non-secure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow non-secure 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 non-secure masked flag
This flag is set by hardware when a timestamp non-secure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wakeup timer non-secure masked flag
This flag is set by hardware when the wakeup timer non-secure 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 non-secure masked flag
This flag is set by hardware when the alarm B non-secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A non-secure interrupt occurs.

41.6.22 RTC secure masked interrupt status register (RTC_SMISR)

This register can be globally protected, or each bit of this register can be individually protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

Address offset: 0x58

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 interrupt secure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp secure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow interrupt secure masked flag
This flag is set by hardware when a timestamp secure 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 interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wakeup timer interrupt secure masked flag
This flag is set by hardware when the wakeup timer secure 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 interrupt secure masked flag
This flag is set by hardware when the alarm B secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A interrupt secure masked flag
This flag is set by hardware when the alarm A secure interrupt occurs.

41.6.23 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against non-secure access. Refer to *Section 41.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 41.3.5: RTC privilege protection modes*.

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 ALRBF bit in the RTC_SR register.

41.6.24 RTC register map

Table 307. 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	Res.	Res.
	Reset value																0									0	0	0	0	0	1		
0x10	**RTC_PRER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	**RTC_CR**	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	Res.	Res.	TAMPOE	TAMPTS	ITSE	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	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
0x1C	**RTC_PRIVCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	INITPRIV	CALPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSPRIV	WUTPRIV	ALRBPRIV	ALRAPRIV
	Reset value																	0	0	0										0	0	0	0
0x20	**RTC_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DECPROT	INITDPROT	CALDPROT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSDPROT	WUTDPROT	ALRBDPROT	ALRADPROT
	Reset value																	1	1	1										1	1	1	1
0x24	**RTC_WPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_ CALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	**RTC_SHIFTR**	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 307. 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
0x30	**RTC_TSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	**RTC_TSDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	**RTC_TSSSR**	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
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
0x58	**RTC_SMISR**	Res.	Res.	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 2.3 on page 87* for the register boundary addresses.

42 Tamper and backup registers (TAMP)

42.1 Introduction

32 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. 8 tamper pins and 5 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, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

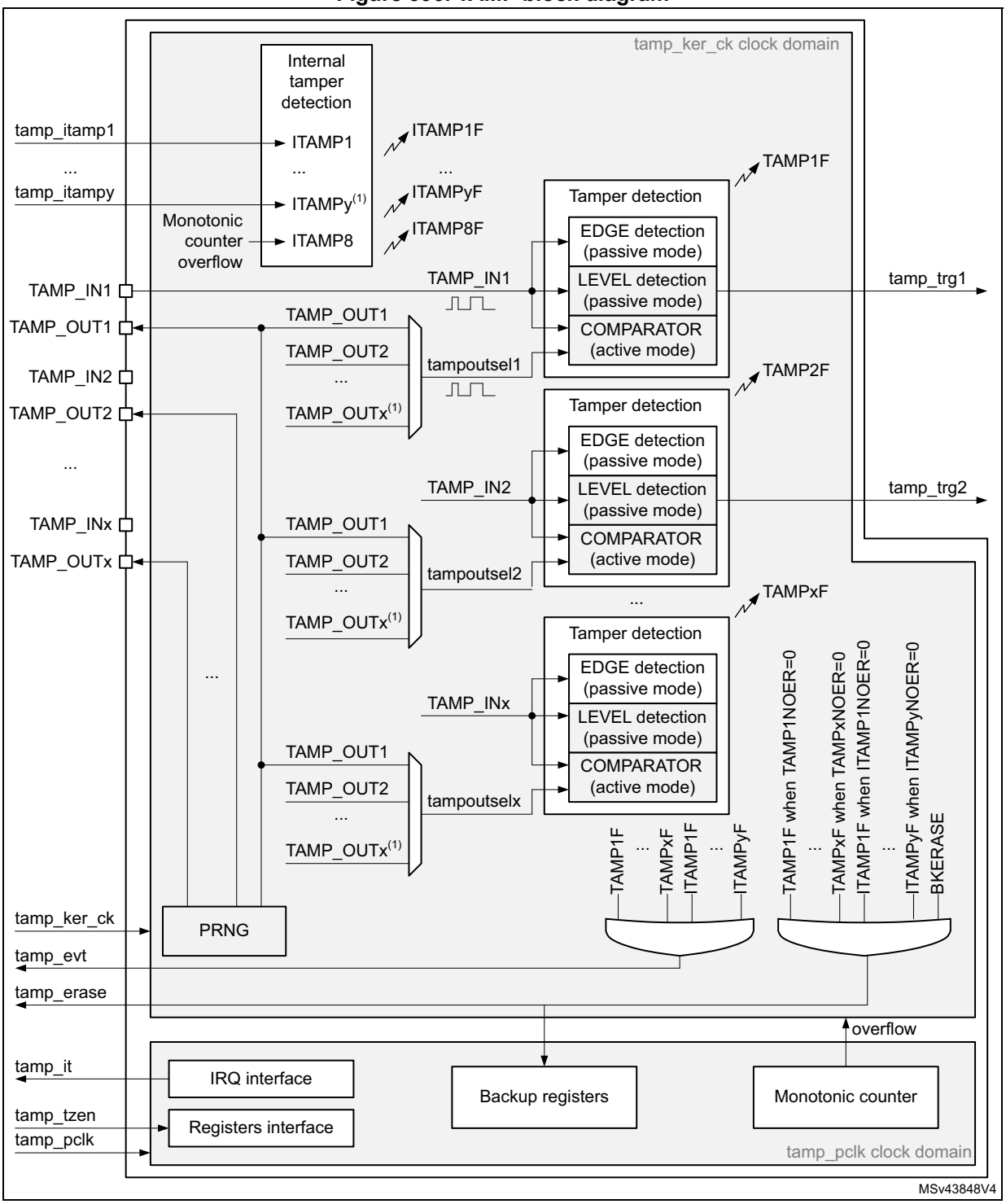
42.2 TAMP main features

- 32 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.
- 8 external tamper detection events.
  - Each external event can be configured to be active or passive.
  - External passive tampers with configurable filter and internal pull-up.
- 5 internal tamper events.
- Any tamper detection can generate a RTC timestamp event.
- Any tamper detection can erase the backup registers, SRAM2, ICACHE, PKA SRAM and cryptographic peripherals.
- TrustZone support:
  - Tamper secure or non-secure configuration.
  - Backup registers configuration in 3 configurable-size areas:
    1 read/write secure area.
    1 write secure/read non-secure area.
    1 read/write non-secure area.
- Monotonic counter.

42.3 TAMP functional description

42.3.1 TAMP block diagram

Figure 396. TAMP block diagram



1. The number of external and internal tampers depends on products.

42.3.2 TAMP pins and internal signals

Table 308. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin
TAMP_OUTx (x = pin index)	Output	Tamper output pin

Table 309. 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_tzen	Input	TAMP TrustZone enabled
tamp_evt	Output	Tamper event detection (internal or external) The tamp_evt is used to generate a RTC timestamp event
tamp_erase	Output	Device secrets erase request following either tamper event detection (internal or external) or the software erase request done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 42.5: 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 42.4: TAMP low-power modes* for more details.

Table 310. TAMP interconnection

Signal name	Source/Destination
tamp_tzen	From FLASH option bytes: TZEN
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_erase	The tamp_erase signal is used to erase the device secrets listed hereafter: backup registers, SRAM2, ICACHE, PKA SRAM and cryptographic peripherals
tamp_itamp1	V_{DD} upper voltage threshold monitoring
tamp_itamp2	Temperature monitoring
tamp_itamp3	LSE monitoring

Table 310. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp8[1]	Monotonic counter overflow

1. This signal is generated in the TAMP peripheral.

The TZEN option bit is used to activate TrustZone in the device.

TZEN = 1: TrustZone activated.

TZEN = 0: TrustZone disabled.

When TrustZone is disabled, the APB access to the TAMP registers are non-secure.

42.3.3 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.

42.3.4 TAMP secure protection modes

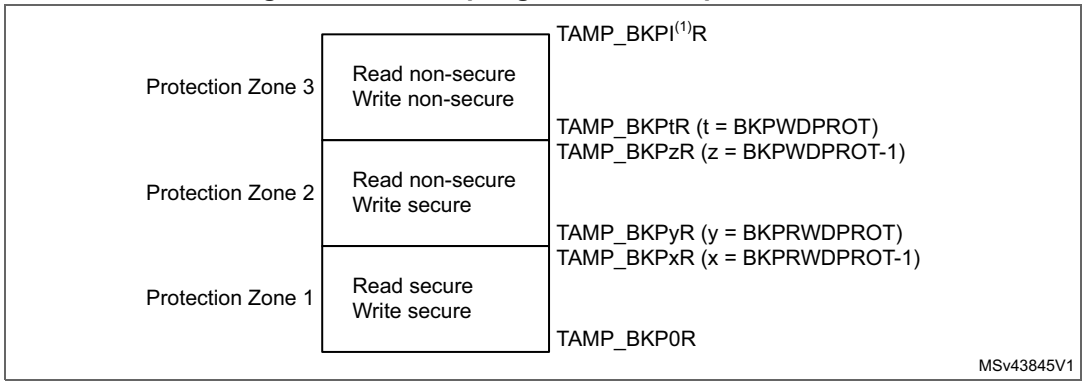
By default after a backup domain power-on reset, all TAMP registers can be read or written in both secure and non-secure modes, except for the TAMP secure mode control register (TAMP_SMCR) which can be written in secure mode only when TZEN = 1. The TAMP protection configuration is not affected by a system reset.

When the TAMPDPROT bit is cleared in the TAMP_SMCR register:

- Writing the TAMP registers is possible only in secure mode, except for the backup registers which have their own protection setting.
- Reading TAMP_SMCR, TAMP_PRIVCR and TAMP_MISR is always possible in secure and non-secure modes. All the other TAMP registers can be read only in secure mode, except for the backup registers which have their own protection setting.

The backup registers protection is configured thanks to BKPRWDPROT[7:0] and BKPWDPROT[7:0] (refer to *Figure 397* below):

Figure 397. Backup registers secure protections



1. I= last backup register index

In case TZEN =1, the bits BKPWPRIV and BKPRWPRIV in the TAMP_PRIVCR can be written only in secure mode.

A non-secure access to a secure-protected register is denied:

- There is no bus error generated.
- A notification is generated through a flag/interrupt in the TZIC (TrustZone interrupt controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

42.3.5 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and non-privileged modes, except for the TAMP privilege mode control register (TAMP_PRIVCR) which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

When the TAMPPRIV bit is set in the TAMP_PRIVCR register:

- Writing the TAMP registers is possible only in privilege mode, except for the backup registers which have their own protection setting.
- Reading TAMP_SMCR, TAMP_PRIVCR is always possible in privilege and non-privilege modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers which have their own protection setting.

The backup registers protection is configured thanks to BKPRWDPROT[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRWDPROT[7:0], BKPWDPROT[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 397*). In case TZEN = 0, BKPRWDPROT[7:0] and BKPWDPROT[7:0] in TAMP_SMCR register can be read and written with non-secure access.

A non-privileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

42.3.6 Tamper detection

The tamper detection can be configured for the following purposes:

- erase the backup registers and the SRAMs listed in *Table 310: TAMP interconnection* (default configuration)
- generate an interrupt, capable to wakeup from Stop and Standby mode
- generate a hardware trigger for the low-power timers

TAMP backup registers

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs except if the TAMPxNOER bit is set, or if the TAMPxMSK is set in the TAMP_CR2 register, or if the ITAMPxNOER bit is set in the TAMP_CR3 register.

The backup registers and the device secrets erased by tamp_erase signal (refer to *Table 310: TAMP interconnection*) can be reset by software by setting the BKERASE bit in the TAMP_CR2 register.

Note: The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.

Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

When TAMPxMSK is cleared:

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck_apre cycles when TAMPFLT differs from 0x0 (level detection with filtering)
- 3 ck_apre cycles when TAMPTS = 1 (timestamp on tamper event)
- No latency when TAMPFLT = 0x0 (edge detection) and TAMPTS = 0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

When TAMPxMSK is set:

A new tamper occurring on the same pin cannot be detected during the latency described above and 2.5 ck_rtc additional cycles.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit in cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration allows the low-power timers in Stop mode to be trigged automatically, without requiring the system wakeup to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in the *Level detection with filtering on tamper inputs (passive mode)* mode.

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a

normal timestamp event occurs. The affected tamper flag register TAMPxF is set in the TAMP_SR at the same time that TSF or TSOVF is set in the RTC_SR.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge/high level or a falling edge/low level is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When using the edge detection, it is recommended to check by software the tamper pin level just after enabling the tamper detection (by reading the GPIO registers), and before writing sensitive values in the backup registers, to ensure that an active edge did not occur before enabling the tamper event detection.
When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0) the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCR register.

Every two CK_ATPER cycles (CK_ATPER = 2^{ATPER} x CK_ATPRE = 2^{ATPER} x $2^{ATCKSEL}$ RTCCLK), TAMP_OUTy output pin provides a value provided by a pseudo random number

generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.

PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.
- When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods

The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

Table 311. Minimum ATPER value

Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, taking into account feedback delay. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 00 and ATOSEL2 = 00.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

The pseudo-random generator must be initially and periodically fed with a new seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

Active tamper initialization

Here is the software procedure to initialize the active tampers after system reset:

Read INITS in TAMP_ATOR register.

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in the TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in RTC_ATOR. Backup registers are then protected by active tamper.
- If INITS = 0x1 (initialization already done):
  No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the RTC APB clock.
- In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in the TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in RTC_ATOR. Backup registers are then protected by active tamper.

42.4 TAMP low-power modes

Table 312. 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 and active tamper modes 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 and active tamper modes 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 and active tamper modes which remain active only when the clock source is LSE. Tamper events cause the device to exit the Shutdown mode.

Table 313. TAMP pins functionality over modes

Pin name	Functional in all low-power modes	Functional in V_{BAT} mode
TAMP_IN[8:1]	yes	only TAMP_IN1,2,3
TAMP_OUT[8:1]	yes	only TAMP_OUT2

42.5 TAMP interrupts

The interrupt channel is set in the interrupt status register or in the secure interrupt status register depending on its secure mode (TAMPDPROT) configuration. The interrupt output or the secure interrupt output is also activated.

Table 314. Non-secure 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 and TAMPDPROT=1	Write 1 in CTAMPxF	Yes	Yes(4)	Yes(5)
	Internal tamper y(3)	ITAMPyF	TAMPyIE and TAMPDPROT=1	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, or in case of active 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, or in case of active tamper mode, wakeup from Shutdown modes is possible only when the TAMP clock source is LSE.

Table 315. Secure 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_S	Tamper x(3)	TAMPxF	TAMPxIE and TAMPDPROT=0	Write 1 in CTAMPxF	Yes	Yes(4)	Yes(5)
	Internal tamper y(3)	ITAMPyF	TAMPyIE and TAMPDPROT=0	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_SMISR register.
3. The number of tampers and internal tampers events depend on products.
4. In case of level detection with filtering passive tamper mode, or in case of active 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, or in case of active tamper mode, wakeup from Shutdown modes is possible only when the TAMP clock source is LSE.

42.6 TAMP registers

Refer to *Section 1.2 on page 76* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

42.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP*

secure protection modes.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0xFFFF 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8E	Res.	Res.	ITAMP5E	Res.	ITAMP3E	ITAMP2E	ITAMP1E
								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.	TAMP8E	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **ITAMP8E**: Internal tamper 8 enable
0: Internal tamper 8 disabled.
1: Internal tamper 8 enabled.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 **ITAMP2E**: Internal tamper 2 enable
0: Internal tamper 2 disabled.
1: Internal tamper 2 enabled.

Bit 16 **ITAMP1E**: Internal tamper 1 enable
0: Internal tamper 1 disabled.
1: Internal tamper 1 enabled.

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

Bit 7 **TAMP8E**: Tamper detection on TAMP_IN8 enable
0: Tamper detection on TAMP_IN8 is disabled.
1: Tamper detection on TAMP_IN8 is enabled.

Bit 6 **TAMP7E**: Tamper detection on TAMP_IN7 enable
0: Tamper detection on TAMP_IN7 is disabled.
1: Tamper detection on TAMP_IN7 is enabled.

Bit 5 **TAMP6E**: Tamper detection on TAMP_IN6 enable
0: Tamper detection on TAMP_IN6 is disabled.
1: Tamper detection on TAMP_IN6 is enabled.

Bit 4 **TAMP5E**: Tamper detection on TAMP_IN5 enable
0: Tamper detection on TAMP_IN5 is disabled.
1: Tamper detection on TAMP_IN5 is enabled.

Bit 3 **TAMP4E**: Tamper detection on TAMP_IN4 enable
0: Tamper detection on TAMP_IN4 is disabled.
1: Tamper detection on TAMP_IN4 is enabled.

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.

42.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

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
TAMP8 TRG	TAMP7 TRG	TAMP6 TRG	TAMP5 TRG	TAMP4 TRG	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	BK ERASE	Res.	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
rw	rw	rw	rw	rw	rw	rw	rw	w					rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8 NOER	TAMP7 NOER	TAMP6 NOER	TAMP5 NOER	TAMP4 NOER	TAMP3 NOER	TAMP2 NOER	TAMP1 NOER
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMP8TRG**: Active level for tamper 8 input (active mode disabled)

0: If TAMPFLT ≠ 00 Tamper 8 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 8 input rising edge and high level triggers a tamper detection event.

1: If TAMPFLT ≠ 00 Tamper 8 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 8 input falling edge and low level triggers a tamper detection event.

Bit 30 **TAMP7TRG**: Active level for tamper 7 input (active mode disabled)

0: If TAMPFLT ≠ 00 Tamper 7 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 7 input rising edge and high level triggers a tamper detection event.

1: If TAMPFLT ≠ 00 Tamper 7 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 7 input falling edge and low level triggers a tamper detection event.

Bit 29 **TAMP6TRG**: Active level for tamper 6 input (active mode disabled)

0: If TAMPFLT ≠ 00 Tamper 6 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 6 input rising edge and high level triggers a tamper detection event.

1: If TAMPFLT ≠ 00 Tamper 6 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 6 input falling edge and low level triggers a tamper detection event.

Bit 28 **TAMP5TRG**: Active level for tamper 5 input (active mode disabled)

0: If TAMPFLT ≠ 00 Tamper 5 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 5 input rising edge and high level triggers a tamper detection event.

1: If TAMPFLT ≠ 00 Tamper 5 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 5 input falling edge and low level triggers a tamper detection event.

Bit 27 **TAMP4TRG**: Active level for tamper 4 input (active mode disabled)

0: If TAMPFLT ≠ 00 Tamper 4 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 4 input rising edge and high level triggers a tamper detection event.

1: If TAMPFLT ≠ 00 Tamper 4 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 4 input falling edge and low level triggers a tamper detection event.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input

0: If TAMPFLT ≠ 00 Tamper 3 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 3 input rising edge and high level triggers a tamper detection event.

1: If TAMPFLT ≠ 00 Tamper 3 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 3 input falling edge and low level triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input

0: If TAMPFLT ≠ 00 Tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 2 input rising edge and high level triggers a tamper detection event.

1: If TAMPFLT ≠ 00 Tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 2 input falling edge and low level triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input

0: If TAMPFLT ≠ 00 Tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 1 input rising edge and high level triggers a tamper detection event.

1: If TAMPFLT ≠ 00 Tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 1 input falling edge and low level triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers(1) erase

Writing '1' to this bit reset the backup registers(1). Writing 0 has no effect. This bit is always read as 0.

Bits 22:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask

0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.

1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers(1) are not erased.

The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers(1) are not erased.

The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers(1) are not erased.

The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

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

Bit 7 **TAMP8NOER**: Tamper 8 no erase

0: Tamper 8 event erases the backup registers.
1: Tamper 8 event does not erase the backup registers(1).

Bit 6 **TAMP7NOER**: Tamper 7 no erase

0: Tamper 7 event erases the backup registers.
1: Tamper 7 event does not erase the backup registers(1).

Bit 5 **TAMP6NOER**: Tamper 6 no erase

0: Tamper 6 event erases the backup registers.
1: Tamper 6 event does not erase the backup registers(1).

Bit 4 **TAMP5NOER**: Tamper 5 no erase

0: Tamper 5 event erases the backup registers.
1: Tamper 5 event does not erase the backup registers(1).

Bit 3 **TAMP4NOER**: Tamper 4 no erase

0: Tamper 4 event erases the backup registers.
1: Tamper 4 event does not erase the backup registers(1).

Bit 2 **TAMP3NOER**: Tamper 3 no erase
0: Tamper 3 event erases the backup registers.
1: Tamper 3 event does not erase the backup registers(1).

Bit 1 **TAMP2NOER**: Tamper 2 no erase
0: Tamper 2 event erases the backup registers.
1: Tamper 2 event does not erase the backup registers(1).

Bit 0 **TAMP1NOER**: Tamper 1 no erase
0: Tamper 1 event erases the backup registers.
1: Tamper 1 event does not erase the backup registers(1).

1. The device secrets erased by tamp_erase signal (refer to *Table 310: TAMP interconnection*).

42.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8NOER	Res.	Res.	ITAMP5NOER	Res.	ITAMP3NOER	ITAMP2NOER	ITAMP1NOER
								rw			rw		rw	rw	rw

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

Bit 7 **ITAMP8NOER**: Internal Tamper 8 no erase
0: Internal Tamper 8 event erases the backup registers.
1: Internal Tamper 8 event does not erase the backup registers(1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ITAMP5NOER**: Internal Tamper 5 no erase
0: Internal Tamper 5 event erases the backup registers.
1: Internal Tamper 5 event does not erase the backup registers(1).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ITAMP3NOER**: Internal Tamper 3 no erase
0: Internal Tamper 3 event erases the backup registers.
1: Internal Tamper 3 event does not erase the backup registers(1).

Bit 1 **ITAMP2NOER**: Internal Tamper 2 no erase
0: Internal Tamper 2 event erases the backup registers.
1: Internal Tamper 2 event does not erase the backup registers(1).

Bit 0 **ITAMP1NOER**: Internal Tamper 1 no erase
0: Internal Tamper 1 event erases the backup registers.
1: Internal Tamper 1 event does not erase the backup registers(1).

1. and the device secrets erased by tamp_erase signal (refer to *Table 310: TAMP interconnection*).

42.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

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.

42.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x0007 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTEN	ATO SHARE	Res.	Res.	Res.	ATPER[2:0]			Res.	Res.	Res.	Res.	Res.	ATCKSEL[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
ATOSEL4[1:0]		ATOSEL3[1:0]		ATOSEL2[1:0]		ATOSEL1[1:0]		TAMP8 AM	TAMP7 AM	TAMP6 AM	TAMP5 AM	TAMP4 AM	TAMP3 AM	TAMP2 AM	TAMP1 AM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLTEN**: Active tamper filter enable

0: Active tamper filtering disable

1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30 **ATOSHARE**: Active tamper output sharing

0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi

1: Each active tamper input TAMP_INi is compared with TAMPOUTSELx as defined below, with TAMPOUTSELx defined by ATOSELx bits.

TAMP_IN1 is compared with TAMPOUTSEL1

TAMP_IN2 is compared with TAMPOUTSEL2

TAMP_IN3 is compared with TAMPOUTSEL3

TAMP_IN4 is compared with TAMPOUTSEL4

TAMP_IN5 is compared with TAMPOUTSEL5

TAMP_IN6 is compared with TAMPOUTSEL6

TAMP_IN7 is compared with TAMPOUTSEL7

TAMP_IN8 is compared with TAMPOUTSEL8

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:24 **ATPER[2:0]**: Active tamper output change period

The tamper output is changed every CK_ATPER = (2^{ATPER} x CK_ATPRE) cycles. Refer to *Table 311: Minimum ATPER value*.

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

Bits 18:16 **ATCKSEL[2:0]**: Active tamper RTC asynchronous prescaler clock selection

These bits selects the RTC asynchronous prescaler stage output.The selected clock is CK_ATPRE.

$f_{CK\_ATPRE} = f_{RTCCLK} / 2^{ATCKSEL}$ when (PREDIV_A+1) = 128.

000: RTCCLK is selected

001: RTCCLK/2 is selected when (PREDIV_A+1) = 128 (actually selects 1st flip flop output)

010: RTCCLK/4 is selected when (PREDIV_A+1) = 128 (actually selects 2nd flip flop output)

...

111: RTCCLK/128 is selected when (PREDIV_A+1) = 128 (actually selects 7th flip flop output)

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 ck_atpre cycles after all the active tampers are disable.

Bits 15:14 **ATOSEL4[1:0]**: Active tamper shared output 4 selection

00: TAMPOUTSEL4 = TAMP_OUT1

01: TAMPOUTSEL4 = TAMP_OUT2

10: TAMPOUTSEL4 = TAMP_OUT3

11: TAMPOUTSEL4 = TAMP_OUT4

The selected output must be available in the package pinout.

Bits 13:12 **ATOSEL3[1:0]**: Active tamper shared output 3 selection

00: TAMPOUTSEL3 = TAMP_OUT1

01: TAMPOUTSEL3 = TAMP_OUT2

10: TAMPOUTSEL3 = TAMP_OUT3

11: TAMPOUTSEL3 = TAMP_OUT4

The selected output must be available in the package pinout

Bits 11:10 **ATOSEL2[1:0]**: Active tamper shared output 2 selection
00: TAMPOUTSEL2 = TAMP_OUT1
01: TAMPOUTSEL2 = TAMP_OUT2
10: TAMPOUTSEL2 = TAMP_OUT3
11: TAMPOUTSEL2 = TAMP_OUT4
The selected output must be available in the package pinout

Bits 9:8 **ATOSEL1[1:0]**: Active tamper shared output 1 selection
00: TAMPOUTSEL1 = TAMP_OUT1
01: TAMPOUTSEL1 = TAMP_OUT2
10: TAMPOUTSEL1 = TAMP_OUT3
11: TAMPOUTSEL1 = TAMP_OUT4
The selected output must be available in the package pinout

Bit 7 **TAMP8AM**: Tamper 8 active mode
0: Tamper 8 detection mode is passive.
1: Tamper 8 detection mode is active.

Bit 6 **TAMP7AM**: Tamper 7 active mode
0: Tamper 7 detection mode is passive.
1: Tamper 7 detection mode is active.

Bit 5 **TAMP6AM**: Tamper 6 active mode
0: Tamper 6 detection mode is passive.
1: Tamper 6 detection mode is active.

Bit 4 **TAMP5AM**: Tamper 5 active mode
0: Tamper 5 detection mode is passive.
1: Tamper 5 detection mode is active.

Bit 3 **TAMP4AM**: Tamper 4 active mode
0: Tamper 4 detection mode is passive.
1: Tamper 4 detection mode is active.

Bit 2 **TAMP3AM**: Tamper 3 active mode
0: Tamper 3 detection mode is passive.
1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode
0: Tamper 2 detection mode is passive.
1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode
0: Tamper 1 detection mode is passive.
1: Tamper 1 detection mode is active.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

42.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x14

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
SEED[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
SEED[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SEED[31:0]**: Pseudo-random generator seed value

This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.

42.6.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset 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.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
r	r							r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **INITS**: Active tamper initialization status

This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is cleared when the active tampers are disabled.

Bit 14 **SEEDF**: Seed running flag

This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB cock must not be switched off as long as SEEDF is set.

Bits 13:8 Reserved, must be kept at reset value.

Bits 7:0 **PRNG[7:0]**: Pseudo-random generator value

This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.

This field can only be read when the APB is in secure mode.

42.6.8 TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x1C

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
ATOSEL8[2:0]			ATOSEL7[2:0]			ATOSEL6[2:0]			ATOSEL5[2:0]			ATOSEL4[2:0]			ATOSEL3[2]
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
ATOSEL3[1:0]		ATOSEL2[2:0]			ATOSEL1[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:29 **ATOSEL8[2:0]**: Active tamper shared output 8 selection
000: TAMPOUTSEL8 = TAMP_OUT1
001: TAMPOUTSEL8 = TAMP_OUT2
010: TAMPOUTSEL8 = TAMP_OUT3
011: TAMPOUTSEL8 = TAMP_OUT4
100: TAMPOUTSEL8 = TAMP_OUT5
101: TAMPOUTSEL8 = TAMP_OUT6
110: TAMPOUTSEL8 = TAMP_OUT7
111: TAMPOUTSEL8 = TAMP_OUT8
The selected output must be available in the package pinout.

Bits 28:26 **ATOSEL7[2:0]**: Active tamper shared output 7 selection
000: TAMPOUTSEL7 = TAMP_OUT1
001: TAMPOUTSEL7 = TAMP_OUT2
010: TAMPOUTSEL7 = TAMP_OUT3
011: TAMPOUTSEL7 = TAMP_OUT4
100: TAMPOUTSEL7 = TAMP_OUT5
101: TAMPOUTSEL7 = TAMP_OUT6
110: TAMPOUTSEL7 = TAMP_OUT7
111: TAMPOUTSEL7 = TAMP_OUT8
The selected output must be available in the package pinout.

Bits 25:23 **ATOSEL6[2:0]**: Active tamper shared output 6 selection
000: TAMPOUTSEL6 = TAMP_OUT1
001: TAMPOUTSEL6 = TAMP_OUT2
010: TAMPOUTSEL6 = TAMP_OUT3
011: TAMPOUTSEL6 = TAMP_OUT4
100: TAMPOUTSEL6 = TAMP_OUT5
101: TAMPOUTSEL6 = TAMP_OUT6
110: TAMPOUTSEL6 = TAMP_OUT7
111: TAMPOUTSEL6 = TAMP_OUT8
The selected output must be available in the package pinout.

Bits 22:20 **ATOSEL5[2:0]**: Active tamper shared output 5 selection
000: TAMPOUTSEL5 = TAMP_OUT1
001: TAMPOUTSEL5 = TAMP_OUT2
010: TAMPOUTSEL5 = TAMP_OUT3
011: TAMPOUTSEL5 = TAMP_OUT4
100: TAMPOUTSEL5 = TAMP_OUT5
101: TAMPOUTSEL5 = TAMP_OUT6
110: TAMPOUTSEL5 = TAMP_OUT7
111: TAMPOUTSEL5 = TAMP_OUT8
The selected output must be available in the package pinout.

Bits 19:17 **ATOSEL4[2:0]**: Active tamper shared output 4 selection
000: TAMPOUTSEL4 = TAMP_OUT1
001: TAMPOUTSEL4 = TAMP_OUT2
010: TAMPOUTSEL4 = TAMP_OUT3
011: TAMPOUTSEL4 = TAMP_OUT4
100: TAMPOUTSEL4 = TAMP_OUT5
101: TAMPOUTSEL4 = TAMP_OUT6
110: TAMPOUTSEL4 = TAMP_OUT7
111: TAMPOUTSEL4 = TAMP_OUT8
The selected output must be available in the package pinout.
Bits 18:17 are the mirror of ATOSEL2[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 16:14 **ATOSEL3[2:0]**: Active tamper shared output 3 selection
000: TAMPOUTSEL3 = TAMP_OUT1
001: TAMPOUTSEL3 = TAMP_OUT2
010: TAMPOUTSEL3 = TAMP_OUT3
011: TAMPOUTSEL3 = TAMP_OUT4
100: TAMPOUTSEL3 = TAMP_OUT5
101: TAMPOUTSEL3 = TAMP_OUT6
110: TAMPOUTSEL3 = TAMP_OUT7
111: TAMPOUTSEL3 = TAMP_OUT8
The selected output must be available in the package pinout.
Bits 15:14 are the mirror of ATOSEL3[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 13:11 **ATOSEL2[2:0]**: Active tamper shared output 2 selection
000: TAMPOUTSEL2 = TAMP_OUT1
001: TAMPOUTSEL2 = TAMP_OUT2
010: TAMPOUTSEL2 = TAMP_OUT3
011: TAMPOUTSEL2 = TAMP_OUT4
100: TAMPOUTSEL2 = TAMP_OUT5
101: TAMPOUTSEL2 = TAMP_OUT6
110: TAMPOUTSEL2 = TAMP_OUT7
111: TAMPOUTSEL2 = TAMP_OUT8
The selected output must be available in the package pinout.
Bits 12:11 are the mirror of ATOSEL2[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 10:8 **ATOSEL1[2:0]**: Active tamper shared output 1 selection
000: TAMPOUTSEL1 = TAMP_OUT1
001: TAMPOUTSEL1 = TAMP_OUT2
010: TAMPOUTSEL1 = TAMP_OUT3
011: TAMPOUTSEL1 = TAMP_OUT4
100: TAMPOUTSEL1 = TAMP_OUT5
101: TAMPOUTSEL1 = TAMP_OUT6
110: TAMPOUTSEL1 = TAMP_OUT7
111: TAMPOUTSEL1 = TAMP_OUT8
The selected output must be available in the package pinout.
Bits 9:8 are the mirror of ATOSEL1[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 7:0 Reserved, must be kept at reset value.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

42.6.9 TAMP secure mode register (TAMP_SMCR)

If TZEN = 1, this register can be written only when the APB access is secure. If TZEN=0, BKPRWDPROT[7:0] and BKPWDPROT[7:0] can be written with non-secure APB access, and TAMPDPROT cannot be written.

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the TAMP_PRIVCR configuration (refer to *Section 42.3.5: TAMP privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x8000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMP DPROT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWDPROT[7:0]							
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.	Res.	Res.	BKPRWDPROT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMPDPROT**: Tamper protection (excluding backup registers)

0: Tamper configuration and interrupt can be written only when the APB access is secure.

1: Tamper configuration and interrupt can be written when the APB access is secure or non-secure.

Note: Refer to Section 42.3.4: TAMP secure protection modes for details on the read protection.

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWDPROT[7:0]**: Backup registers write protection offset

If TZEN=1: backup registers from TAMP_BKPyR (y = BKPRWDPROT, from 0 to 128) to TAMP_BKPzR (z = BKPWDPROT-1, from 0 to 128, BKPWDPROT ≥ BKPRWDPROT) can be written only when the APB is in secure mode. They can be read in secure or non-secure mode. This zone is the protection zone 2.

If TZEN=0: the protection zone 2 can be read and written with non-secure access.

Backup registers from TAMP_BKPtR (t = BKPWDPROT, from 0 to 127) can be read or written when the APB is in secure or in non-secure mode. This zone is the protection zone 3.

If BKPWDPROT = 0 or if BKPWDPROT ≤ BKPRWDPROT: none of the backup registers have a secure write access. In these configurations the behavior is equivalent to BKPWDPROT = BKPRWDPROT.

If BKPWPRIV is set, BKPRWDPROT[7:0] and BKPWDPROT[7:0] can be written only in privileged mode.

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

Bits 7:0 **BKPRWDPROT[7:0]**: Backup registers read/write protection offset

If TZEN=1: backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWDPROT-1, from 0 to 128) can be read and written only when the APB is in secure mode. This is the protection zone 1.

If TZEN=0: the protection zone 1 can be read and written with non-secure access.

If BKPRWDPROT = 0 none of the backup registers have a secure read/write access.

If BKPRWPRIV is set, BKPRWDPROT[7:0] can be written only in privileged mode.

42.6.10 TAMP privilege mode control register (TAMP_PRIVCR)

This register can be written only when the APB access is privileged.

When TZEN = 1, this register can be write-protected, or each bit of this register can be individually write-protected against non-secure access depending on the TAMP_SMCR configuration (refer to *Section 42.3.4: TAMP secure protection modes*).

Address offset: 0x24

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
TAMP PRIV	BKP WPRIV	BKPR WPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding backup registers)

0: Tamper configuration and interrupt can be written when the APB is in privileged or non-privileged mode.

1: Tamper configuration and interrupt can be written only when the APB is in privileged mode.

Note: Refer to Section 42.3.5: TAMP privilege protection modes for details on the read protection.

Bit 30 **BKPWPRIV**: Backup registers zone 2 privilege protection

0: Backup registers zone 2 can be written when the APB is in privileged or non-privileged mode.

1: Backup registers zone 2 can be written only when the APB is in privileged mode.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection

0: Backup registers zone 1 can be read and written when the APB is in privileged or non-privileged mode.

1: Backup registers zone 1 can be read and written only when the APB is in privileged mode.

Bits 28:0 Reserved, must be kept at reset value.

42.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8IE	Res.	Res.	ITAMP5IE	Res.	ITAMP3IE	ITAMP2IE	ITAMP1IE
								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.	TAMP8IE	TAMP7IE	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable

0: Internal tamper 8 interrupt disabled.

1: Internal tamper 8 interrupt enabled.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable

0: Internal tamper 5 interrupt disabled.

1: Internal tamper 5 interrupt enabled.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 **ITAMP2IE**: Internal tamper 2 interrupt enable
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16 **ITAMP1IE**: Internal tamper 1 interrupt enable
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled

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

Bit 7 **TAMP8IE**: Tamper 8 interrupt enable
0: Tamper 8 interrupt disabled.
1: Tamper 8 interrupt enabled.

Bit 6 **TAMP7IE**: Tamper 7interrupt enable
0: Tamper 7 interrupt disabled.
1: Tamper 7interrupt enabled.

Bit 5 **TAMP6IE**: Tamper 6 interrupt enable
0: Tamper 6 interrupt disabled.
1: Tamper 6 interrupt enabled.

Bit 4 **TAMP5IE**: Tamper 5 interrupt enable
0: Tamper 5 interrupt disabled.
1: Tamper 5 interrupt enabled.

Bit 3 **TAMP4IE**: Tamper 4 interrupt enable
0: Tamper 4 interrupt disabled.
1: Tamper 4 interrupt enabled.

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.

42.6.12 TAMP status register (TAMP_SR)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5:*

TAMP privilege protection modes.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8F	Res.	Res.	ITAMP5F	Res.	ITAMP3F	ITAMP2F	ITAMP1F
								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.	TAMP8F	TAMP7F	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
								r	r	r	r	r	r	r	r

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

Bit 23 **ITAMP8F**: Internal tamper 8 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5F**: Internal tamper 5 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 **ITAMP2F**: Internal tamper 2 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16 **ITAMP1F**: Internal tamper 1 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

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

Bit 7 **TAMP8F**: TAMP8 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP8 input

Bit 6 **TAMP7F**: TAMP7 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP7 input.

Bit 5 **TAMP6F**: TAMP6 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP6 input.

Bit 4 **TAMP5F**: TAMP5 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP5 input.

Bit 3 **TAMP4F**: TAMP4 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP4 input.

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.

42.6.13 TAMP non-secure masked interrupt status register (TAMP_MISR)

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8MF	Res.	Res.	ITAMP5MF	Res.	ITAMP3MF	ITAMP2MF	ITAMP1MF
								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.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
								r	r	r	r	r	r	r	r

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

Bit 23 **ITAMP8MF**: Internal tamper 8 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 8 non-secure interrupt is raised.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5MF**: Internal tamper 5 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 5 non-secure interrupt is raised.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3MF**: Internal tamper 3 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 non-secure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 2 non-secure interrupt is raised.

Bit 16 **ITAMP1MF**:Internal tamper 1 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 1 non-secure interrupt is raised.

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

Bit 7 **TAMP8MF**: TAMP8 non-secure interrupt masked flag
This flag is set by hardware when the tamper 8 non-secure interrupt is raised.

Bit 6 **TAMP7MF**: TAMP7 non-secure interrupt masked flag
This flag is set by hardware when the tamper 7 non-secure interrupt is raised.

Bit 5 **TAMP6MF**: TAMP6 non-secure interrupt masked flag
This flag is set by hardware when the tamper 6 non-secure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 non-secure interrupt masked flag
This flag is set by hardware when the tamper 5 non-secure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 non-secure interrupt masked flag
This flag is set by hardware when the tamper 4 non-secure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 non-secure interrupt masked flag
This flag is set by hardware when the tamper 3 non-secure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 non-secure interrupt masked flag
This flag is set by hardware when the tamper 2 non-secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 non-secure interrupt masked flag
This flag is set by hardware when the tamper 1 non-secure interrupt is raised.

42.6.14 TAMP secure masked interrupt status register (TAMP_SMISR)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

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.	ITAMP8MF	Res.	Res.	ITAMP5MF	Res.	ITAMP3MF	ITAMP2MF	ITAMP1MF
								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.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
								r	r	r	r	r	r	r	r

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

Bit 23 **ITAMP8MF**: Internal tamper 8 secure interrupt masked flag
This flag is set by hardware when the internal tamper 8 secure interrupt is raised.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5MF**: Internal tamper 5 secure interrupt masked flag
This flag is set by hardware when the internal tamper 5 secure interrupt is raised.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3MF**: Internal tamper 3 secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 secure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 secure interrupt masked flag
This flag is set by hardware when the internal tamper 2 secure interrupt is raised.

Bit 16 **ITAMP1MF**: Internal tamper 1 secure interrupt masked flag
This flag is set by hardware when the internal tamper 1 secure interrupt is raised.

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

Bit 7 **TAMP8MF**: TAMP8 secure interrupt masked flag
This flag is set by hardware when the tamper 8 secure interrupt is raised.

Bit 6 **TAMP7MF**: TAMP7 secure interrupt masked flag
This flag is set by hardware when the tamper 7 secure interrupt is raised.

Bit 5 **TAMP6MF**: TAMP6 secure interrupt masked flag
This flag is set by hardware when the tamper 6 secure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 secure interrupt masked flag
This flag is set by hardware when the tamper 5 secure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 secure interrupt masked flag
This flag is set by hardware when the tamper 4 secure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 secure interrupt masked flag
This flag is set by hardware when the tamper 3 secure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 secure interrupt masked flag
This flag is set by hardware when the tamper 2 secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 secure interrupt masked flag
This flag is set by hardware when the tamper 1 secure interrupt is raised.

42.6.15 TAMP status clear register (TAMP_SCR)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x3C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITAMP8F	Res.	Res.	CITAMP5F	Res.	CITAMP3F	CITAMP2F	CITAMP1F
								w			w		w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP8F	CTAMP7F	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
								w	w	w	w	w	w	w	w

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

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **CITAMP2F**: Clear ITAMP2 detection flag
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16 **CITAMP1F**: Clear ITAMP1 detection flag
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

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

Bit 7 **CTAMP8F**: Clear TAMP8 detection flag
Writing 1 in this bit clears the TAMP8F bit in the TAMP_SR register.

Bit 6 **CTAMP7F**: Clear TAMP7 detection flag
Writing 1 in this bit clears the TAMP7F bit in the TAMP_SR register.

Bit 5 **CTAMP6F**: Clear TAMP6 detection flag
Writing 1 in this bit clears the TAMP6F bit in the TAMP_SR register.

Bit 4 **CTAMP5F**: Clear TAMP5 detection flag
Writing 1 in this bit clears the TAMP5F bit in the TAMP_SR register.

Bit 3 **CTAMP4F**: Clear TAMP4 detection flag
Writing 1 in this bit clears the TAMP4F bit in the TAMP_SR register.

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.

42.6.16 TAMP monotonic counter register (TAMP_COUNTR)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **COUNT[31:0]:**

This register is read-only only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.

42.6.17 TAMP configuration register (TAMP_CFGR)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

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.	Res.	Res.	WUTMONEN	VMONEN	TMONEN	Res.
												rw	rw	rw	

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **WUTMONEN**: voltage and temperature monitor periodic enable by RTC wakeup timer
0: RTC wakeup timer not used to enable voltage and temperature monitors
1: RTC wakeup timer used to enable voltage and temperature monitors

Bit 2 **VMONEN**: voltage monitor enable
0: voltage monitor disable
1: voltage monitor enable

Bit 1 **TMONEN**: temperature monitor enable
0: temperature monitor disable
1: temperature monitor enable

Bit 0 Reserved, must be kept at reset value.

42.6.18 TAMP backup x register (TAMP_BKPxR)

This register can be protected against non-secure access. Refer to *Section 42.3.4: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 42.3.5: TAMP privilege protection modes*.

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.

42.6.19 TAMP register map

Table 316. TAMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TAMP_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8E	Res.	Res.	ITAMP5E	Res.	ITAMP3E	ITAMP2E	ITAMP1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8E	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
	Reset value									1			1		1	1	1									0	0	0	0	0	0	0	0
0x04	**TAMP_CR2**	TAMP8TRG	TAMP7TRG	TAMP6TRG	TAMP5TRG	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	Res.	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8NOER	TAMP7NOER	TAMP6NOER	TAMP5NOER	TAMP4NOER	TAMP3NOER	TAMP2NOER	TAMP1NOER
	Reset value	0	0	0	0	0	0	0	0	0					0	0	0									0	0	0	0	0	0	0	0
0x08	**TAMP_CR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8NOER	Res.	Res.	ITAMP5NOER	Res.	ITAMP3NOER	ITAMP2NOER	ITAMP1NOER
	Reset value																									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
0x10	**TAMP_ATCR1**	FLTEN	ATOSHARE	Res.	Res.	Res.	AT-PER[2:0]			Res.	Res.	Res.	Res.	Res.	ATCK-SEL[2:0]			ATO SEL4 [1:0]		ATO SEL3 [1:0]		ATO SEL2 [1:0]		ATO SEL1 [1:0]		TAMP8AM	TAMP7AM	TAMP6AM	TAMP5AM	TAMP4AM	TAMP3AM	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0						1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**TAMP_ ATSEEDR**	SEED[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**TAMP_ATOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
	Reset value																	0	0							0	0	0	0	0	0	0	0
0x1C	**TAMP_ATCR2**	ATO SEL8 [2:0]			ATO SEL7 [2:0]			ATO SEL6 [2:0]			ATO SEL5 [2:0]			ATO SEL4 [2:0]			ATO SEL3 [2:0]			ATO SEL2 [2:0]			ATO SEL1 [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								
0x20	**TAMP_SMCR**	TAMPDPROT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWDPROT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWDPROT[7:0]							
	Reset value	1								0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x240	**TAMP_ PRIVCR**	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	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	0																													

Table 316. TAMP 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
0x2C	**TAMP_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8IE	Res.	Res.	ITAMP5IE	Res.	ITAMP3IE	ITAMP2IE	ITAMP1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8IE	TAMP7IE	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value									0			0		0	0	0									0	0	0	0	0	0	0	0
0x30	**TAMP_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8F	Res.	Res.	ITAMP5F	Res.	ITAMP3F	ITAMP2F	ITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8F	TAMP7F	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
	Reset value									0			0		0	0	0									0	0	0	0	0	0	0	0
0x34	**TAMP_MISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8MF	Res.	Res.	ITAMP5MF	Res.	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value									0			0		0	0	0									0	0	0	0	0	0	0	0
0x38	**TAMP_SMISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8MF	Res.	Res.	ITAMP5MF	Res.	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value									0			0		0	0	0									0	0	0	0	0	0	0	0
0x3C	**TAMP_SCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITAMP8F	Res.	Res.	CITAMP5F	Res.	CITAMP3F	CITAMP2F	CITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP8F	CTAMP7F	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value									0			0		0	0	0									0	0	0	0	0	0	0	0
0x40	**TAMP_ COUNTR**	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TAMP_CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUTMONEN	VMONEN	TMONEN	Res.
	Reset value																													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 2.3* for the register boundary addresses.

43 Inter-integrated circuit (I2C) interface

43.1 Introduction

The I^2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I^2C bus. It provides multimaster capability, and controls all I^2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

43.2 I2C main features

- I^2C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available depending on the product implementation (see *Section 43.3: 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.

43.3 I2C implementation

This manual describes the full set of features implemented in I2C peripheral. In the STM32L552xx and STM32L562xx devices I2C1, I2C2, I2C3 and I2C4 implement the full set of features as shown in the following table, with the restriction that only I2C3 can wake up from Stop 2 mode.

Table 317. 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
Wakeup from Stop 2 mode	-	-	X	-
SMBus/PMBus	X	X	X	X

1. X = supported.

43.4 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.

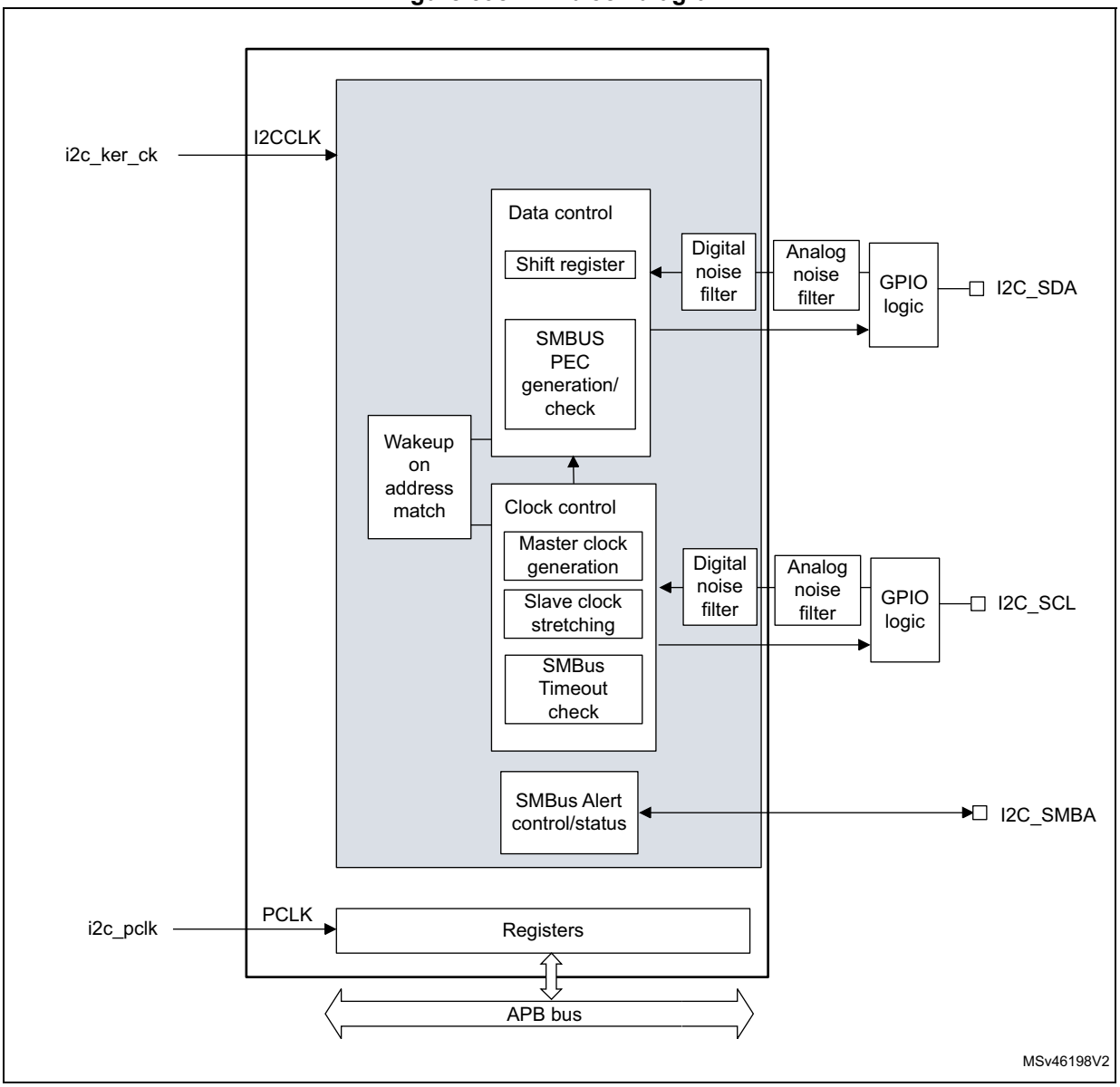
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

43.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 398*.

Figure 398. 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 43.3: I2C implementation.*

43.4.2 I2C pins and internal signals

Table 318. 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 319. 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 333: 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)

43.4.3 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}.

43.4.4 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

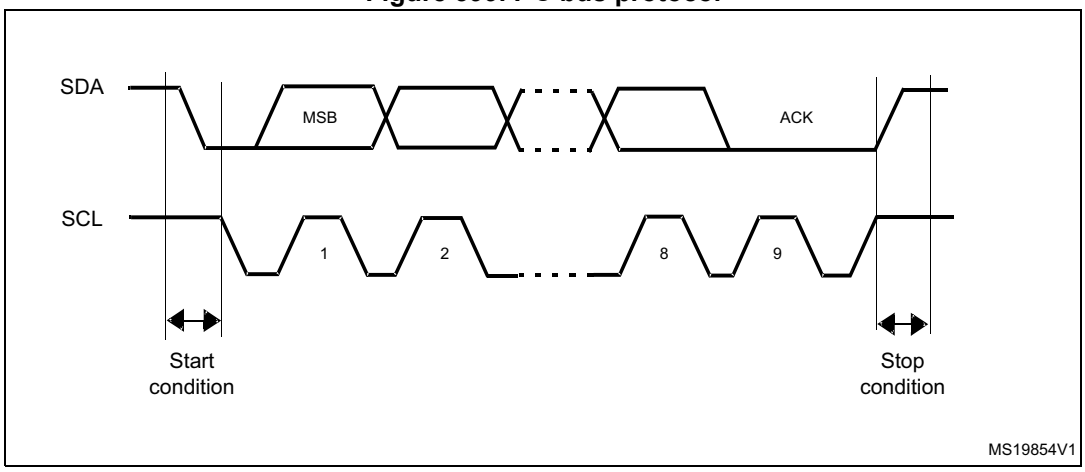
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

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 399. I²C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

43.4.5 I2C initialization

Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller.

Then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE=0), the I^2C performs a software reset. Refer to *Section 43.4.6: 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^2C specification which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows spikes with a programmable length of 1 to 15 I2CCLK periods to be suppressed.

Table 320. 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

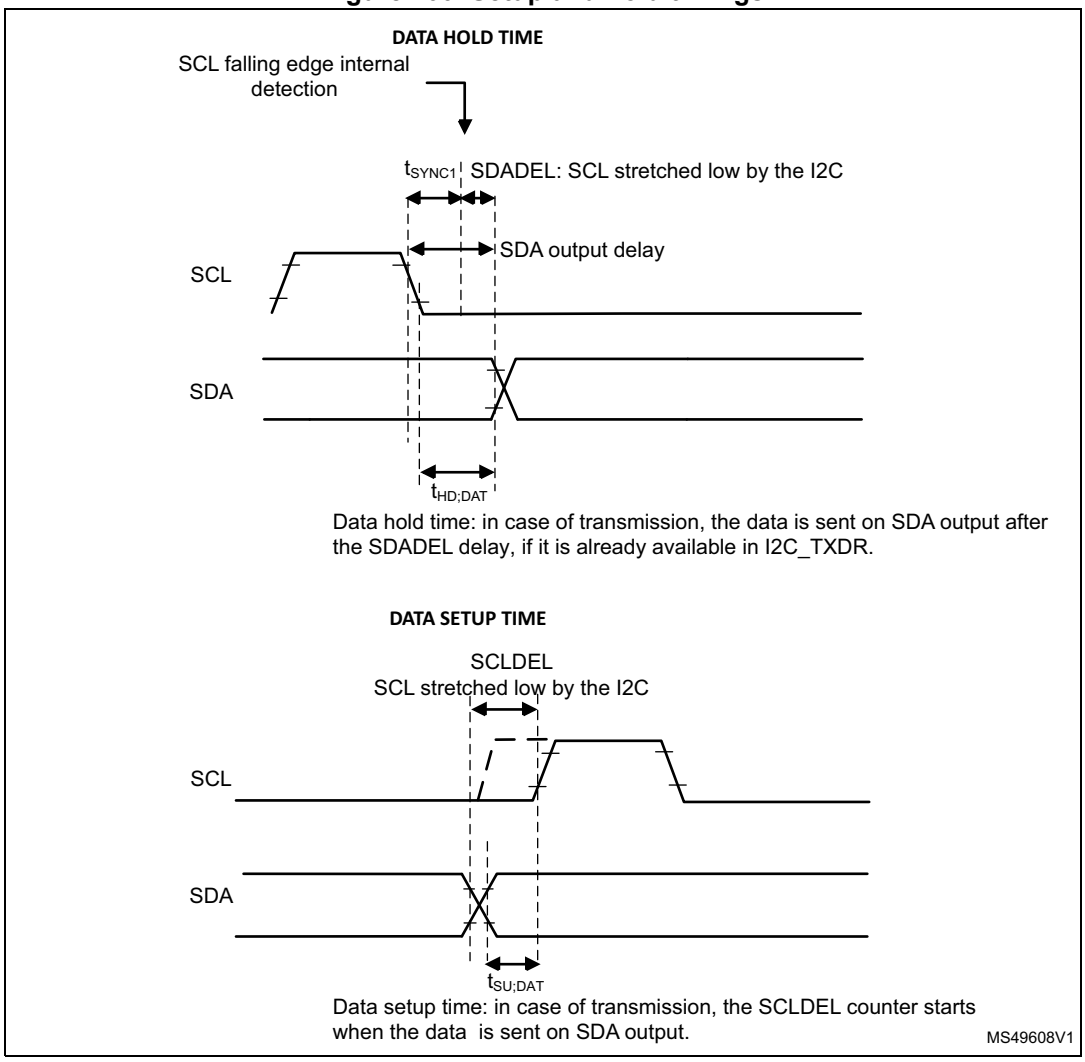
Caution: Changing the filter configuration is not allowed when the I2C is enabled.

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 400. Setup and hold timings



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  T_{SDADEL} impacts the hold time $t_{HD;DAT}$.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC+1) + 1] x t_{I2CCLK} }

t_{SYNC1} duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter: $t_{AF(min)} < t_{AF} < t_{AF(max)}$
- When enabled, input delay brought by the digital filter: t_{DNF} = DNF x t_{I2CCLK}
- Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

$\{t_{f\ (max)} + t_{HD;DAT\ (min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: *$t_{AF(min)}$ / $t_{AF(max)}$ are part of the equation only when the analog filter is enabled. Refer to device datasheet for t_{AF} values.*

The maximum $t_{HD;DAT}$ can be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - 260\ ns - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$.

Note: *This condition can be violated when NOSTRETCH=0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.*

Refer to *Table 321: 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:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC+1)] \times t_{I2CCLK}]\} - 1 <= SCLDEL$

Refer to *Table 321: 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 321. I²C-SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is $t_{SCLL} = (SCLL+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$.
  t_{SCLL} impacts the SCL low time t_{LOW} .
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is $t_{SCLH} = (SCLH+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$. t_{SCLH} impacts the SCL high time t_{HIGH} .

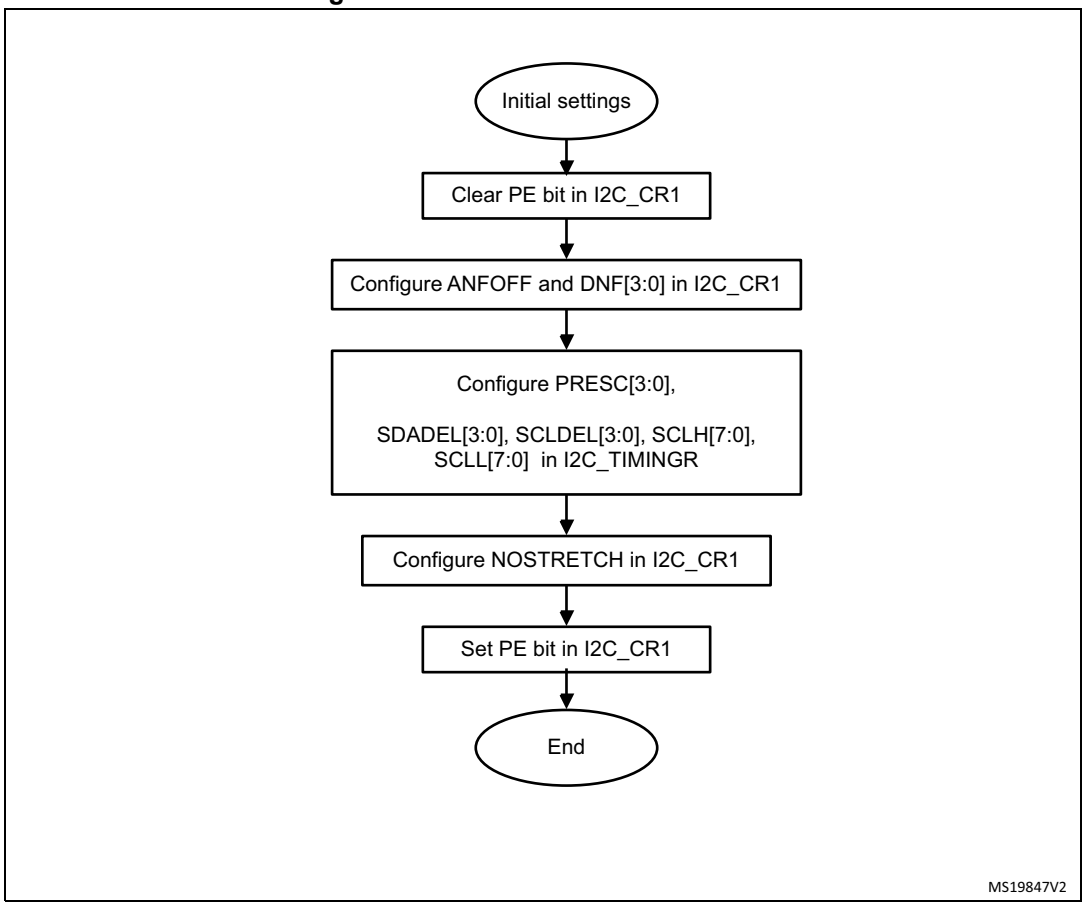
Refer to *I2C master initialization* for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to *I2C slave initialization* for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 401. I2C initialization flowchart



43.4.6 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.

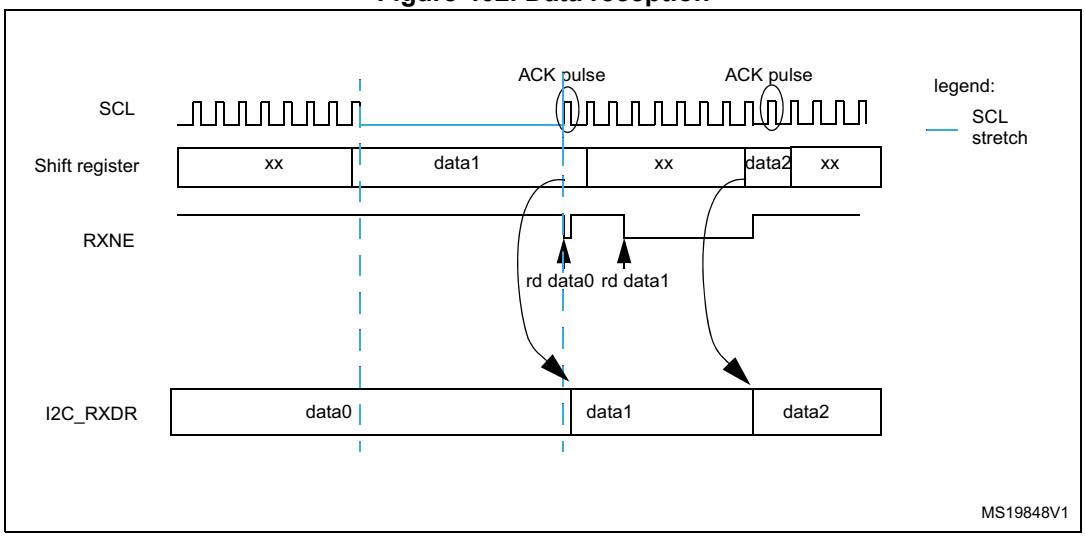
43.4.7 Data transfer

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).

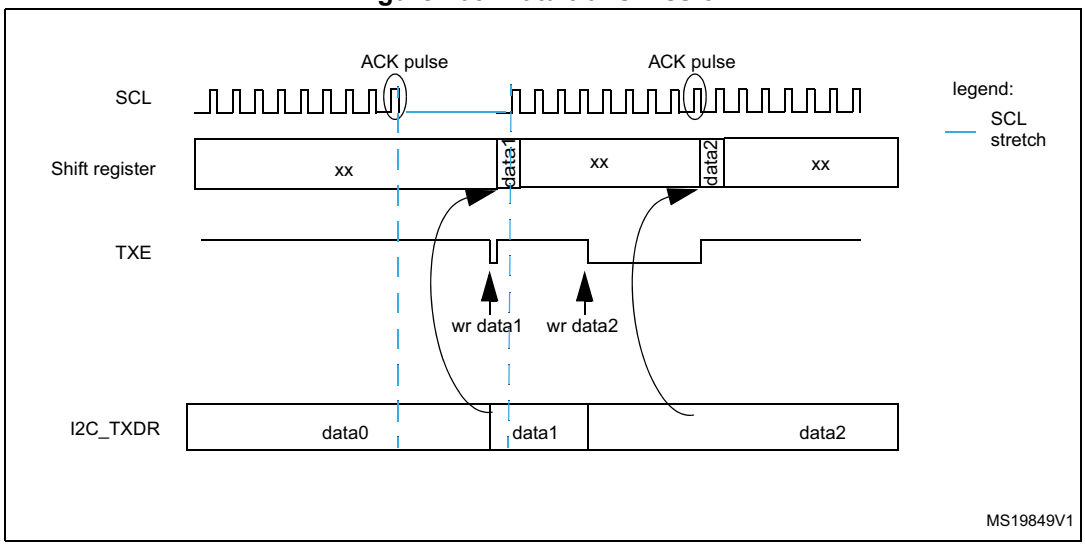
Figure 402. Data reception



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 403. 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 322. 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

43.4.8 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.

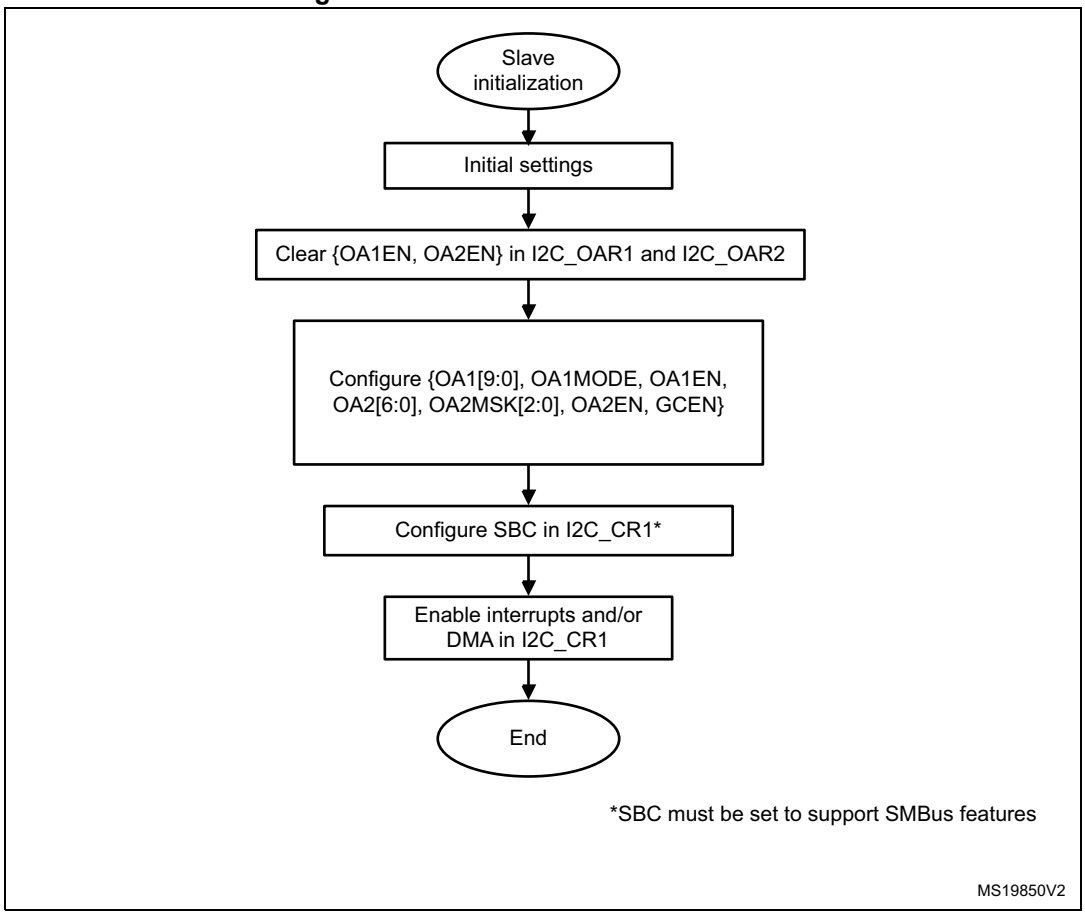
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

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 404. Slave initialization flowchart



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 405. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 0

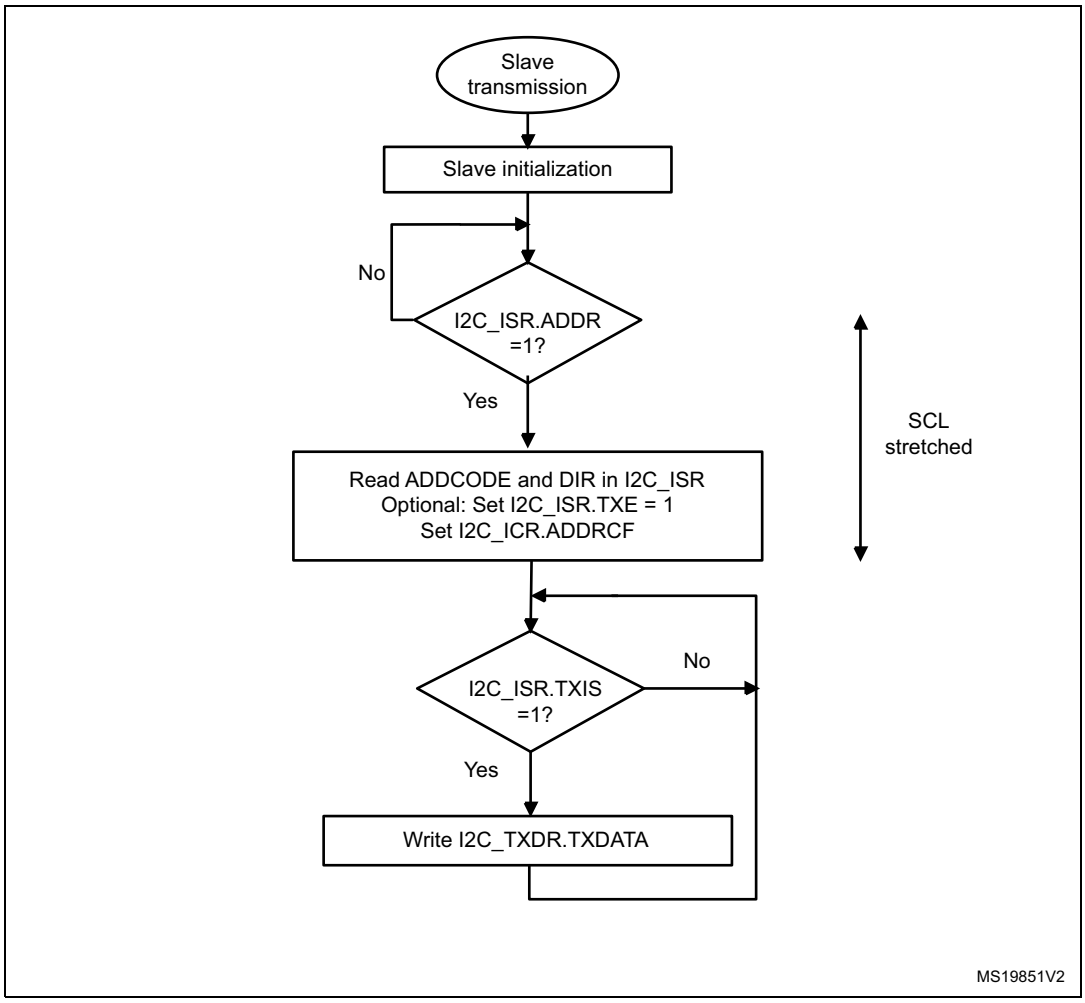


Figure 406. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 1

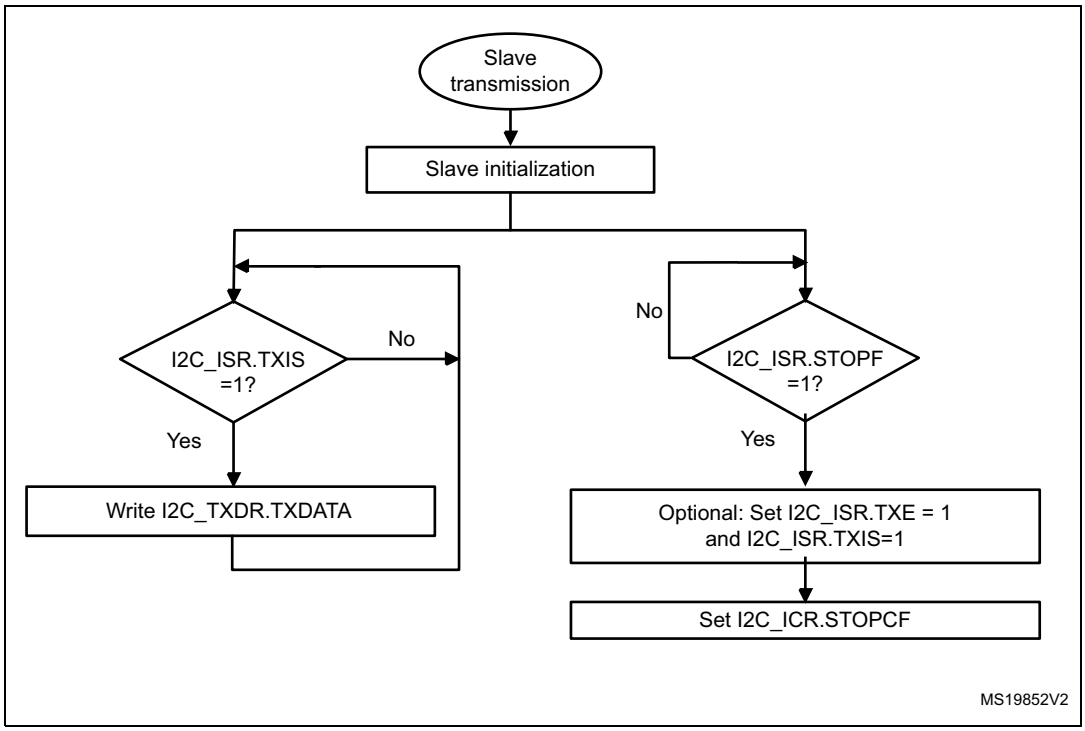
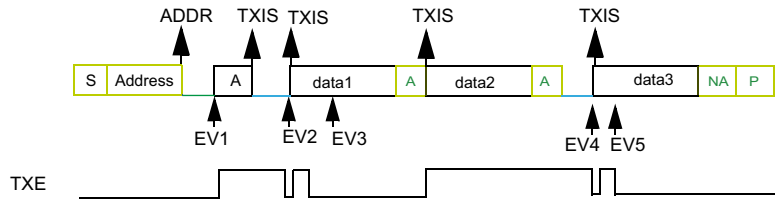


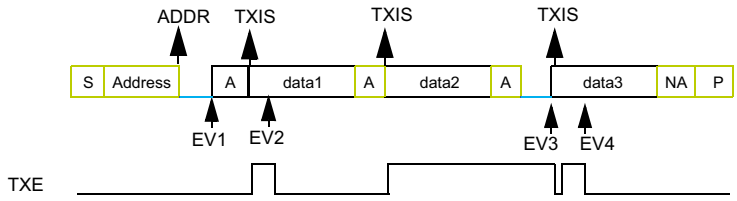
Figure 407. Transfer bus diagrams for I2C slave transmitter

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



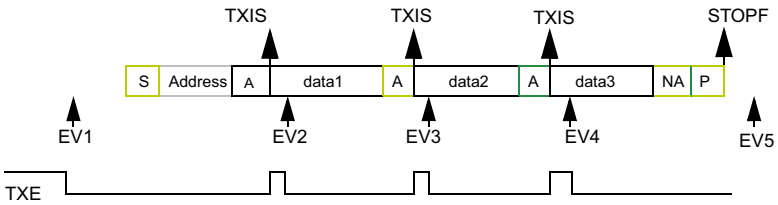
EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:



EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MS19853V2

Slave receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 408. Transfer sequence flowchart for slave receiver with NOSTRETCH=0

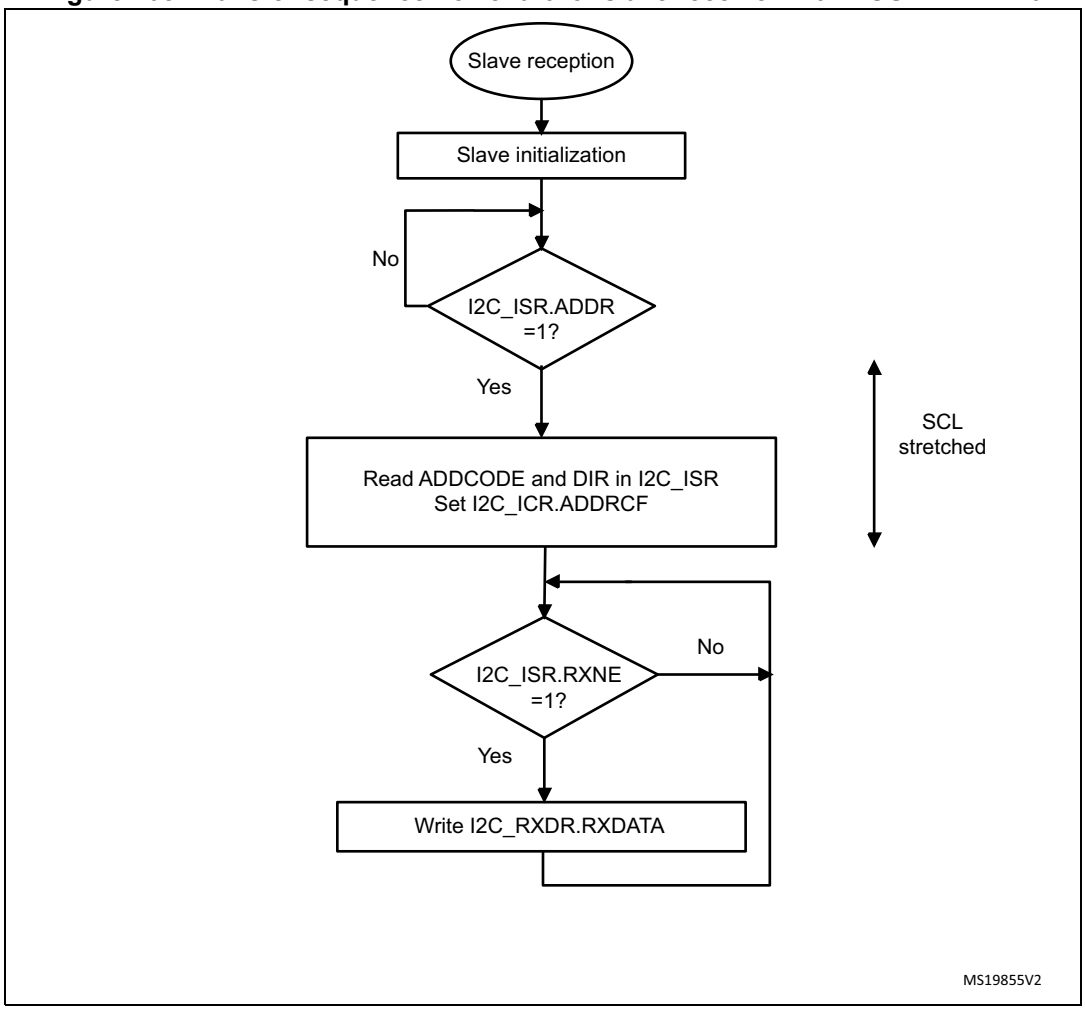


Figure 409. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

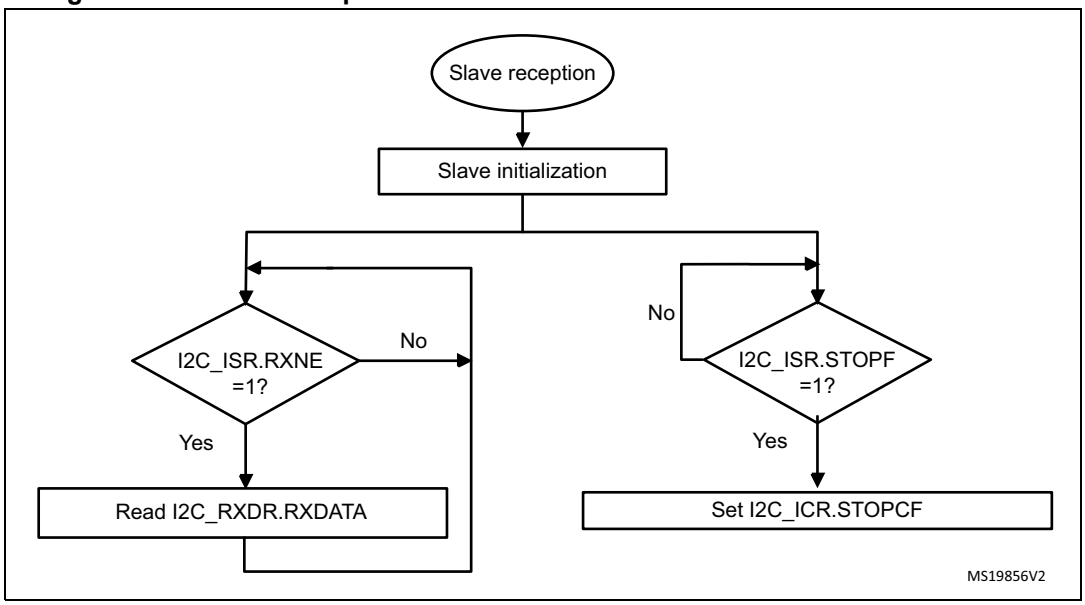
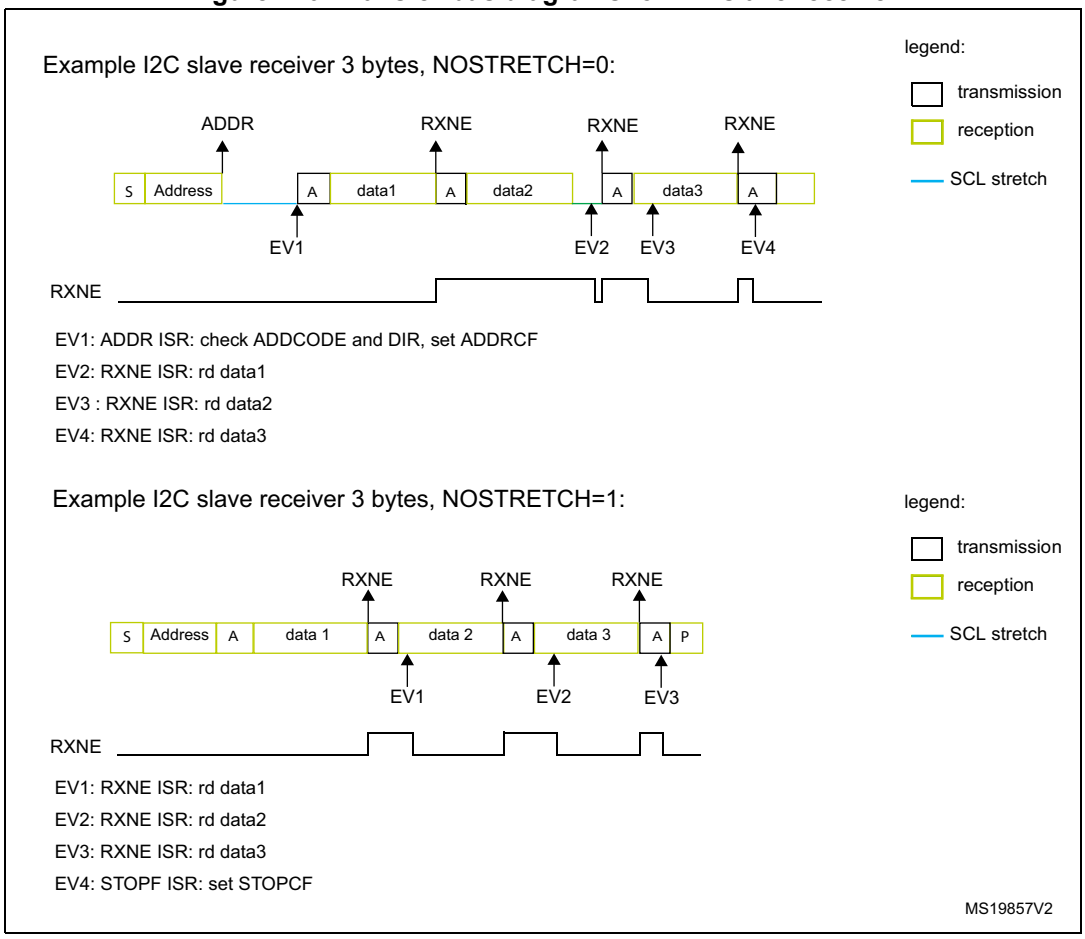


Figure 410. Transfer bus diagrams for I2C slave receiver



43.4.9 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:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$

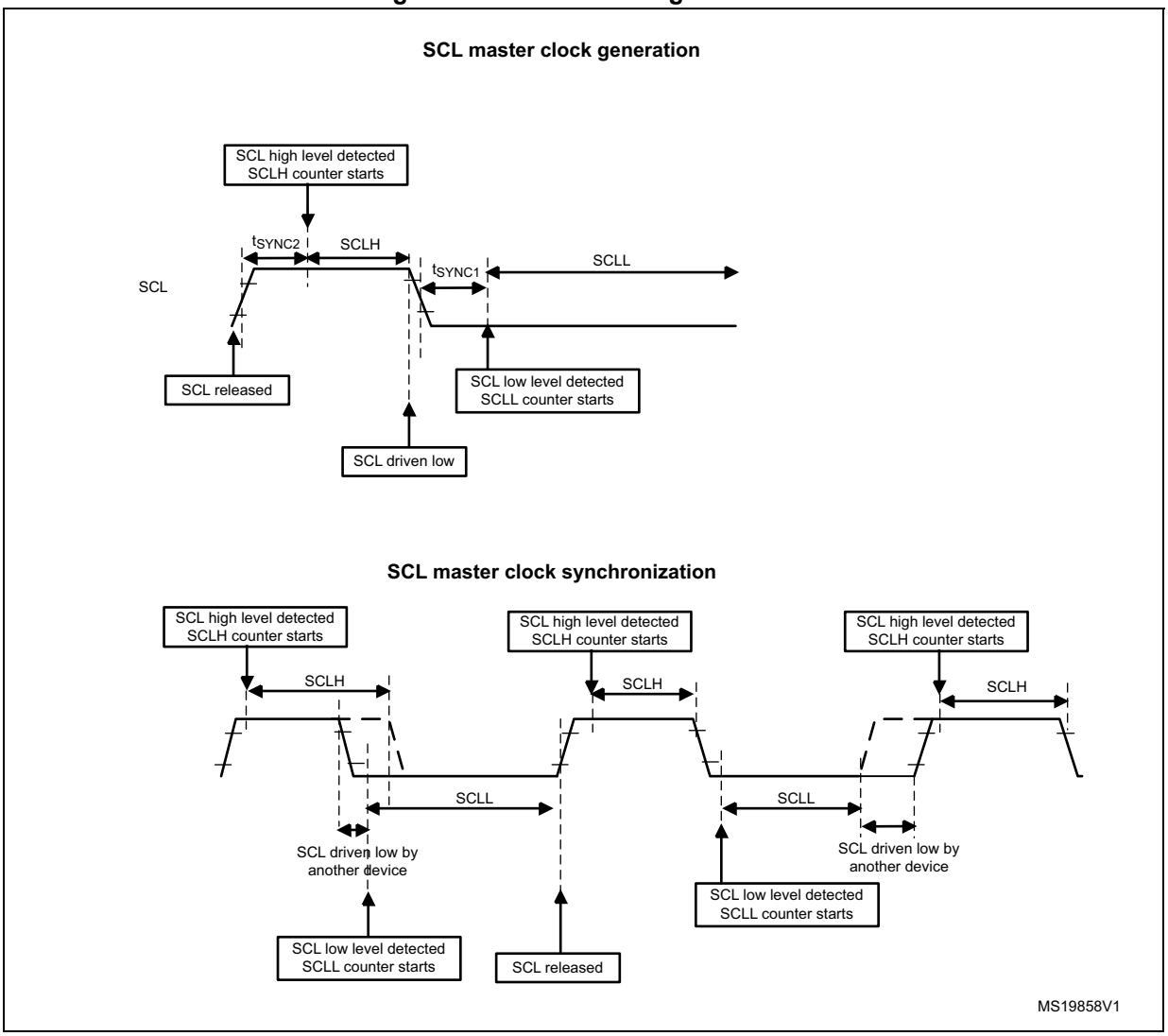
The duration of t_{SYNC1} depends on these parameters:

- 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 411. 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 323. I²C-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 43.4.10: 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}.

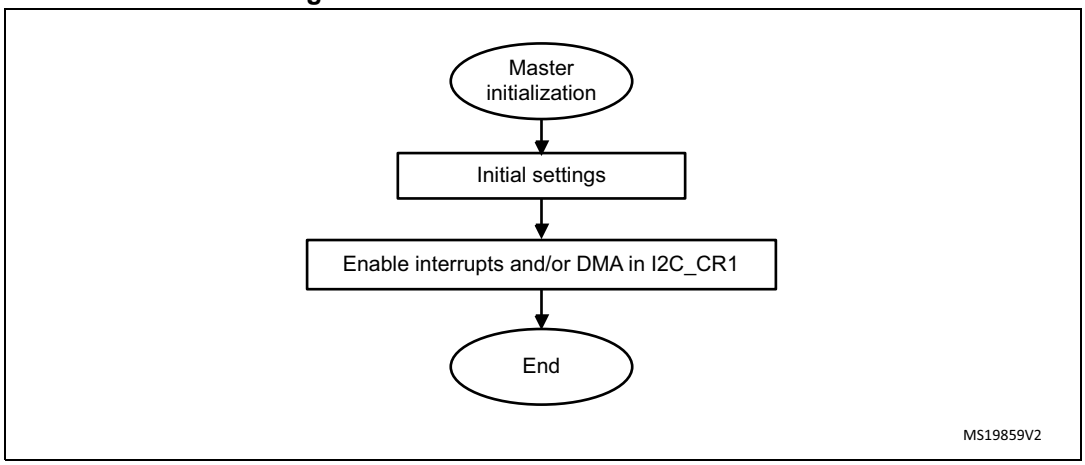
In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

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.

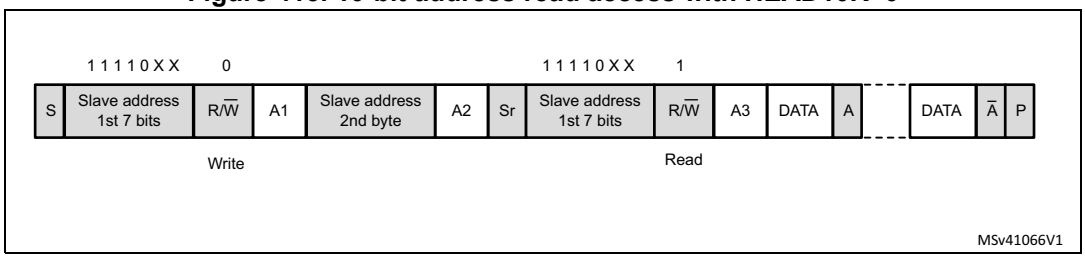
Figure 412. Master initialization flowchart



Initialization of a master receiver addressing a 10-bit address slave

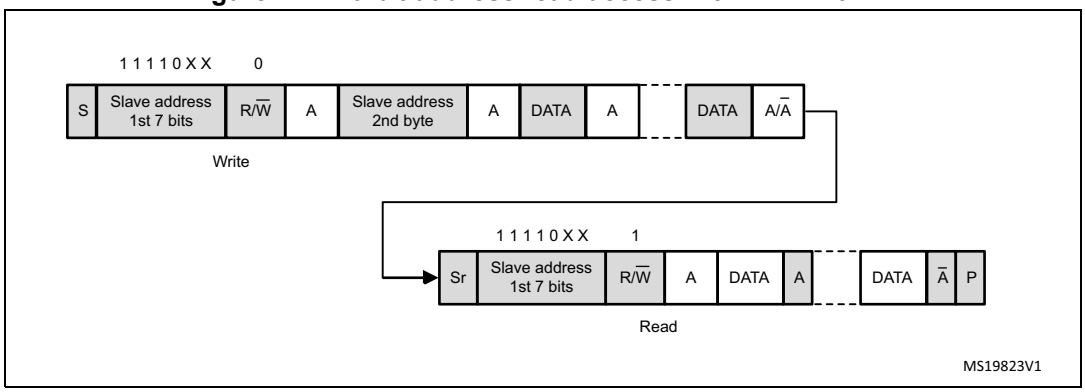
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address 2nd byte + REStart + Slave address 10-bit header Read

Figure 413. 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 414. 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 415. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes

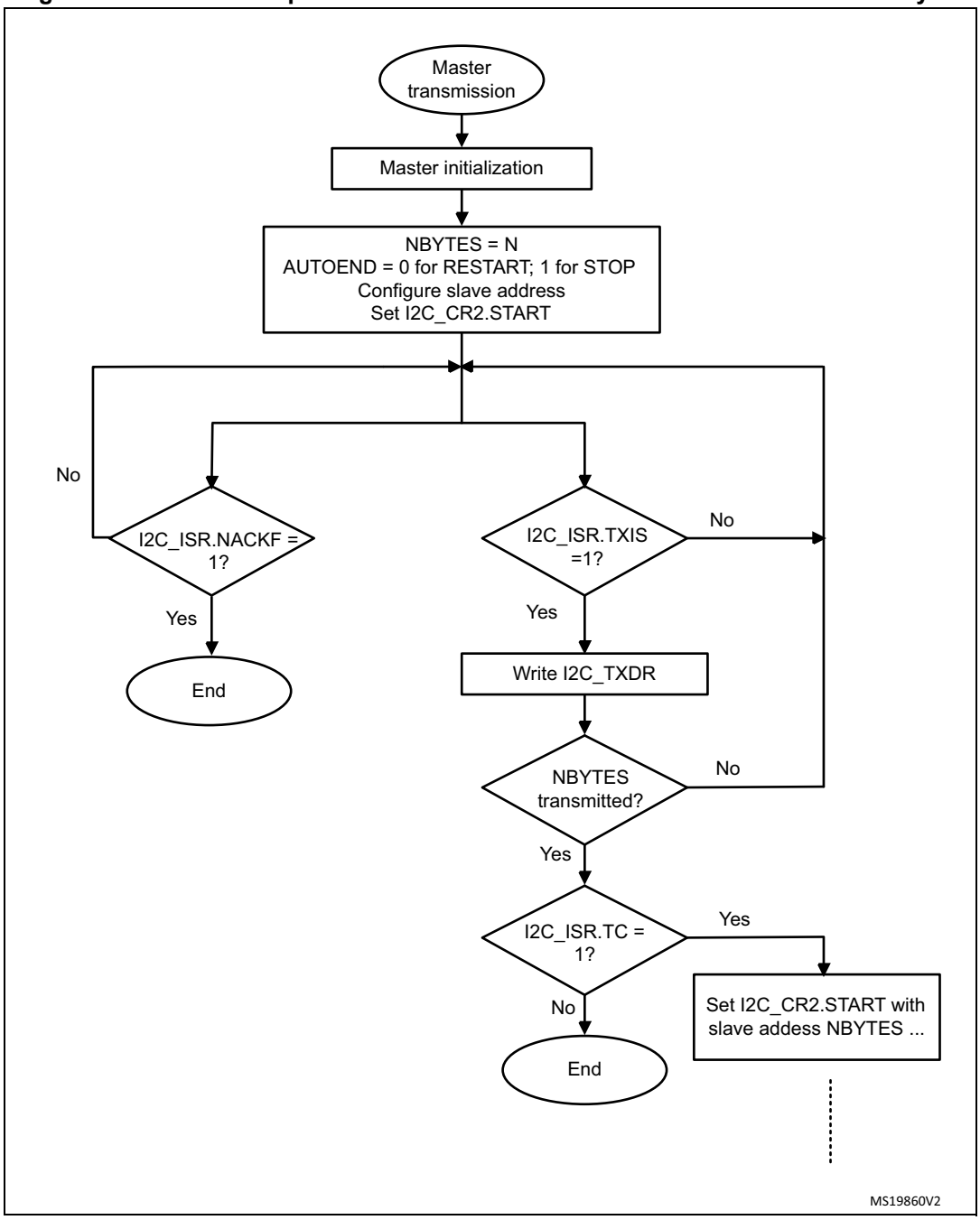


Figure 416. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

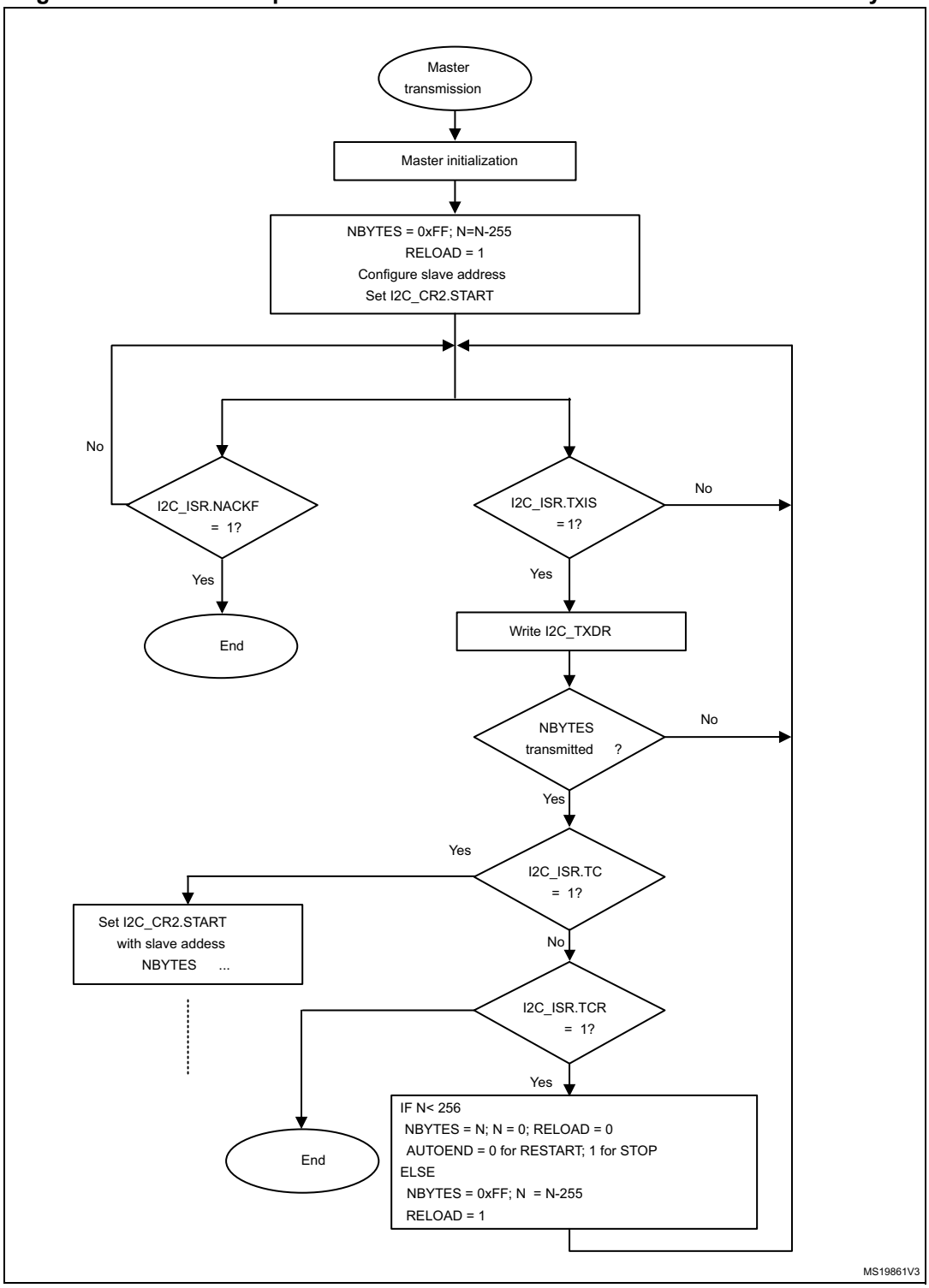
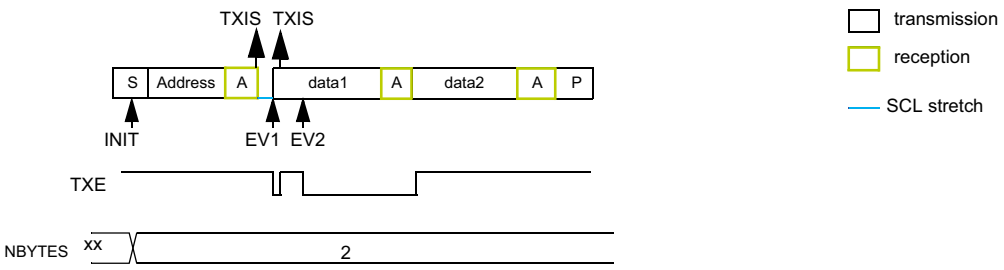


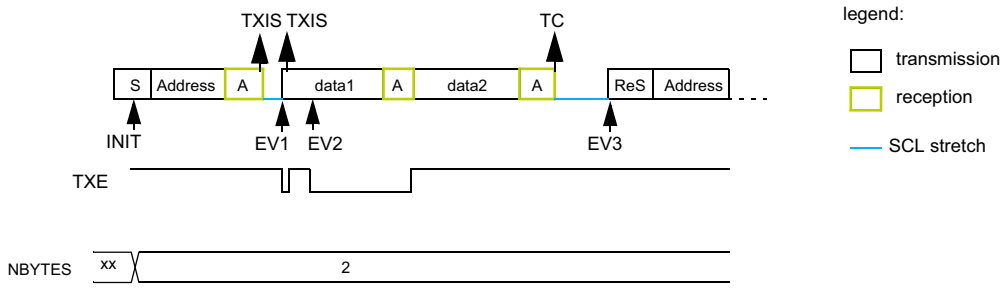
Figure 417. Transfer bus diagrams for I2C master transmitter

Example I2C master transmitter 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19862V2

Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 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 418. Transfer sequence flowchart for I2C master receiver for N≤255 bytes

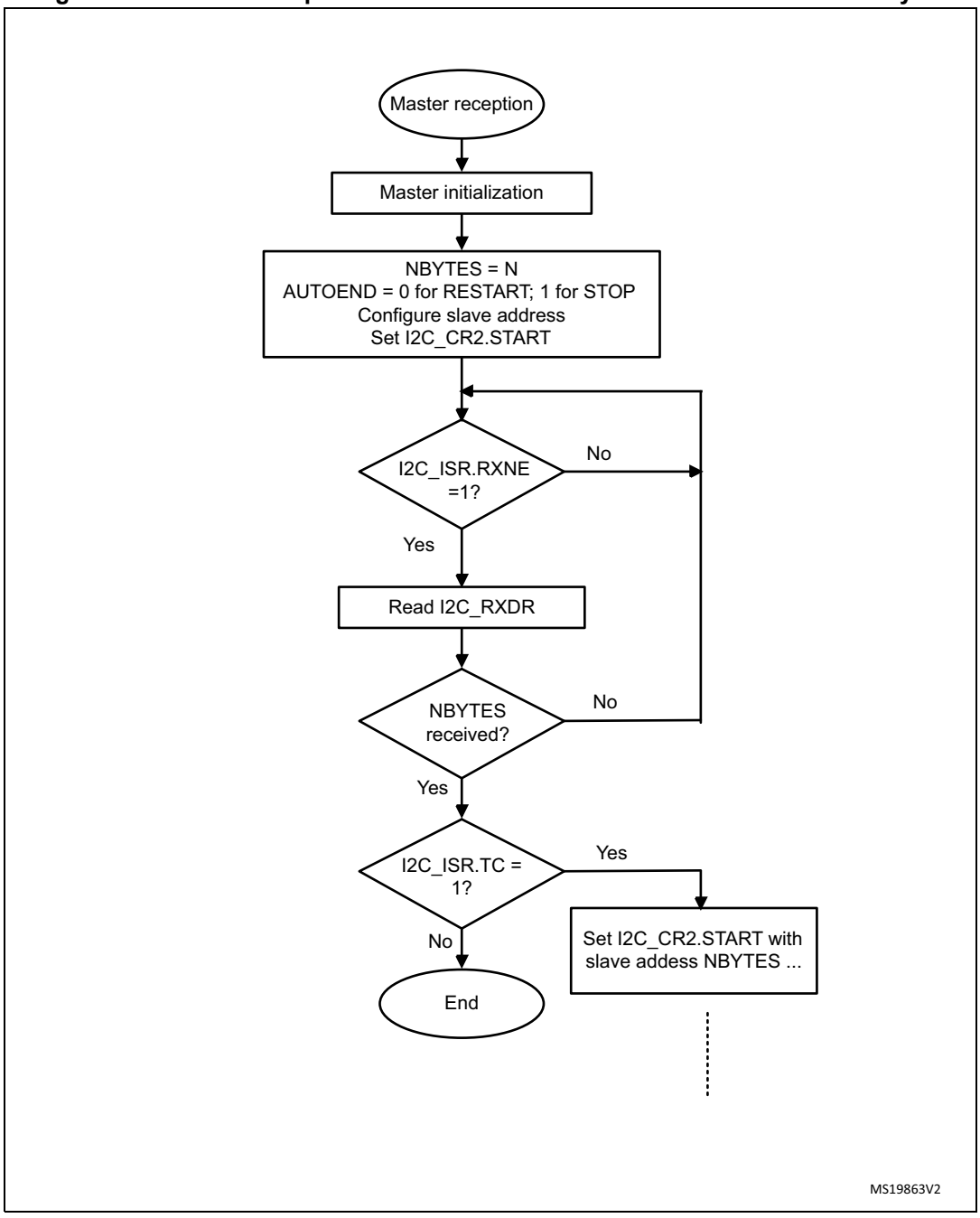


Figure 419. Transfer sequence flowchart for I2C master receiver for N >255 bytes

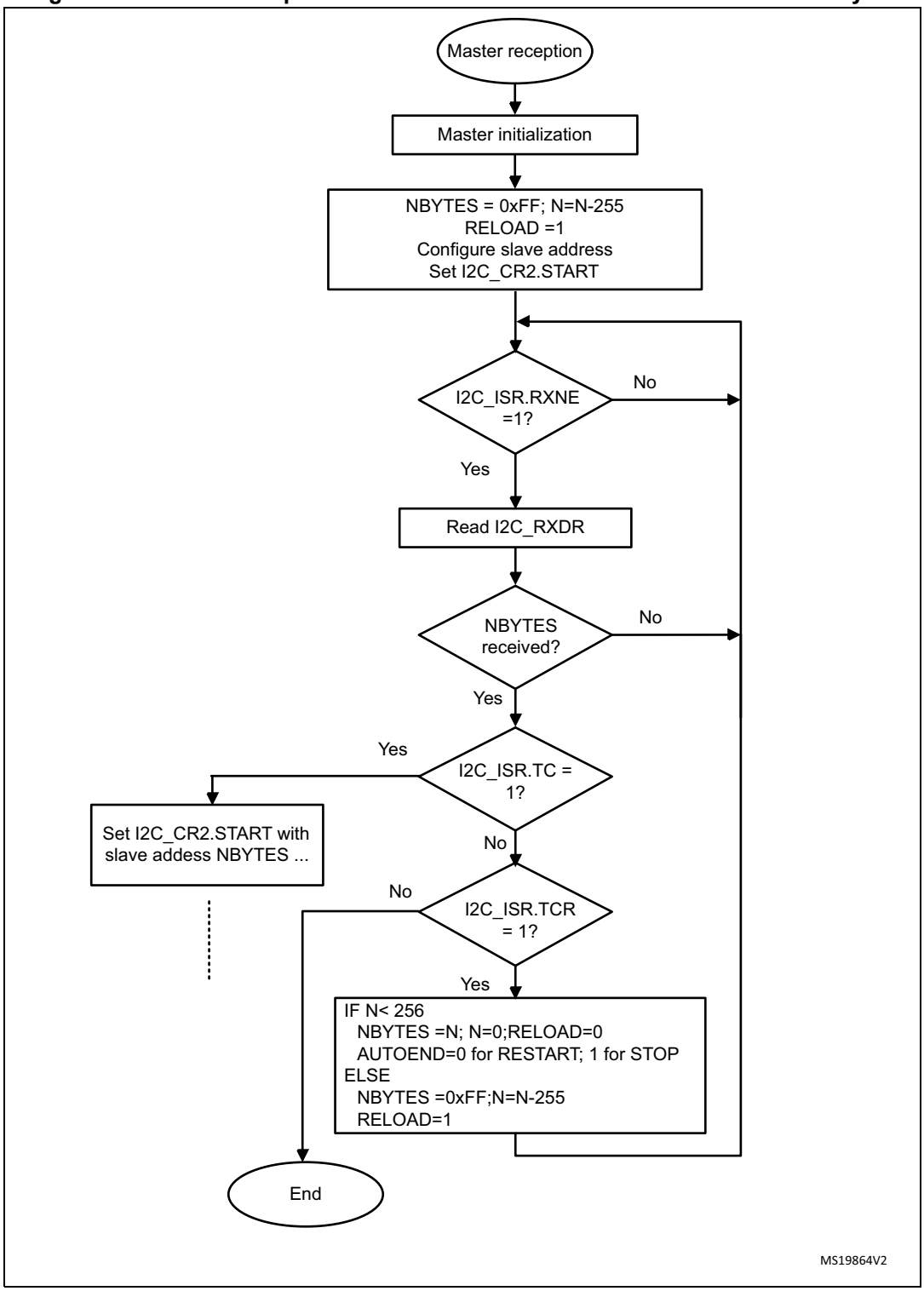
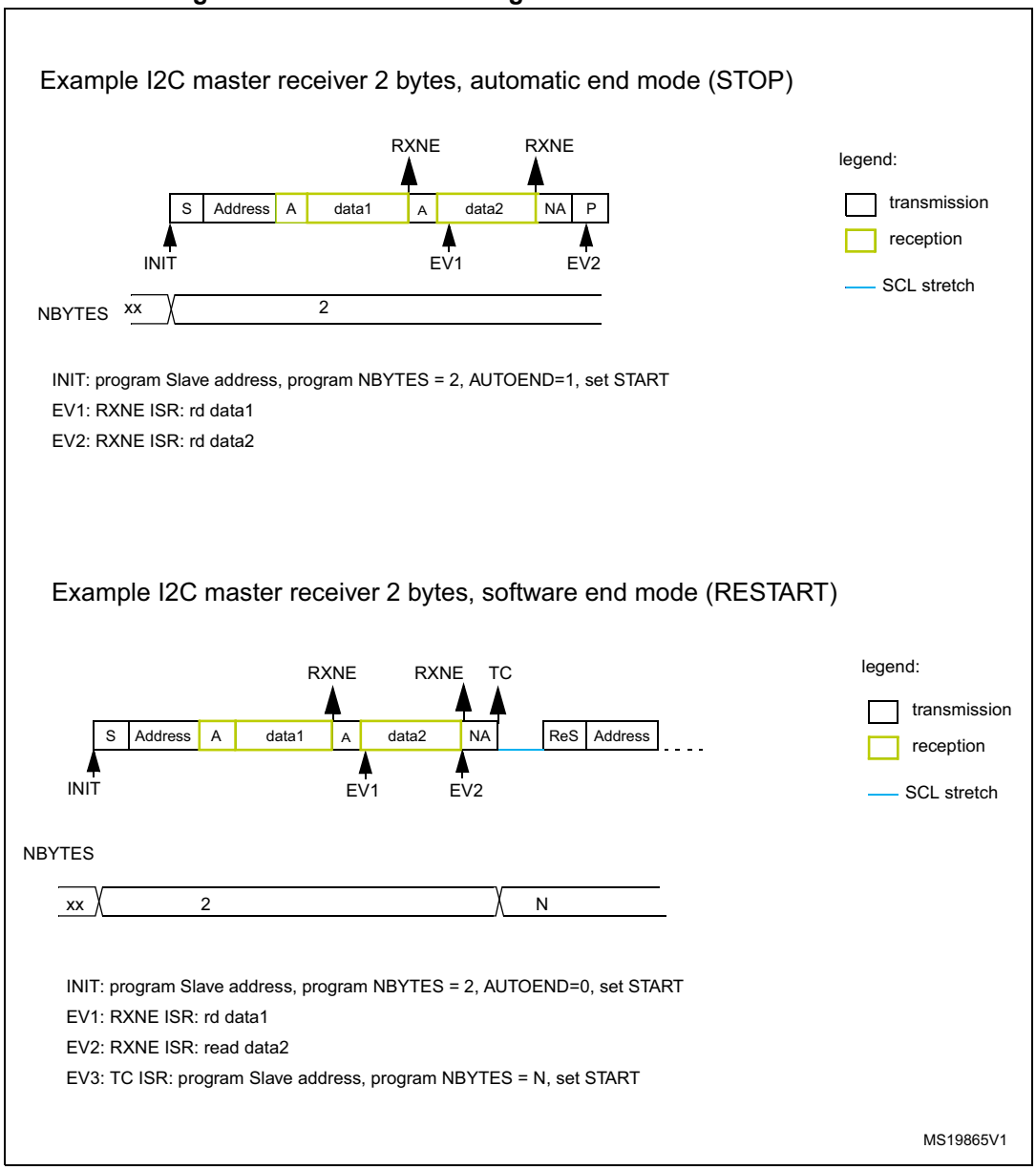


Figure 420. Transfer bus diagrams for I2C master receiver



43.4.10 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I^2C specification. In order to get more accurate configuration values, the STM32CubeMX tool (I2C Configuration window) must be used.

Table 324. 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 325. 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 326. 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

43.4.11 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to *Section 43.3: 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 1503* 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

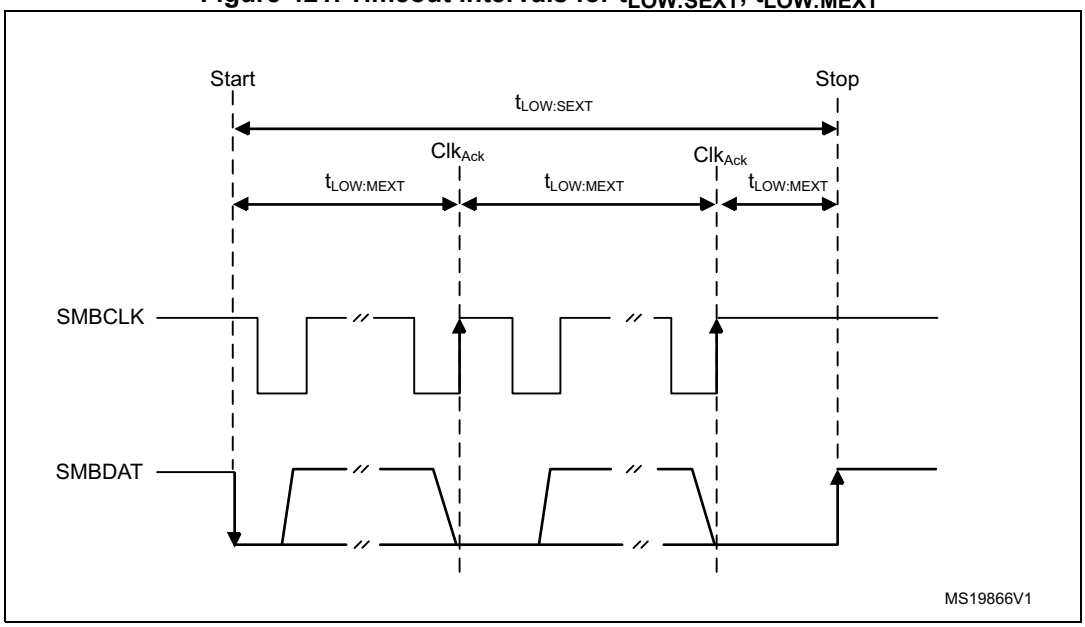
This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification.

Table 327. 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 421. 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 321: 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.

43.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to *Section 43.3: 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 1503* for more details.

Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to *Bus idle detection on page 1526* 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 328. 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 329: 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 1526* section, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 330: 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 331: 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.

43.4.13 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 329. 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 330. 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 331. 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

43.4.14 SMBus slave mode

This section is relevant only when the SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

In addition to I2C slave transfer management (refer to *Section 43.4.8: 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 422. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC

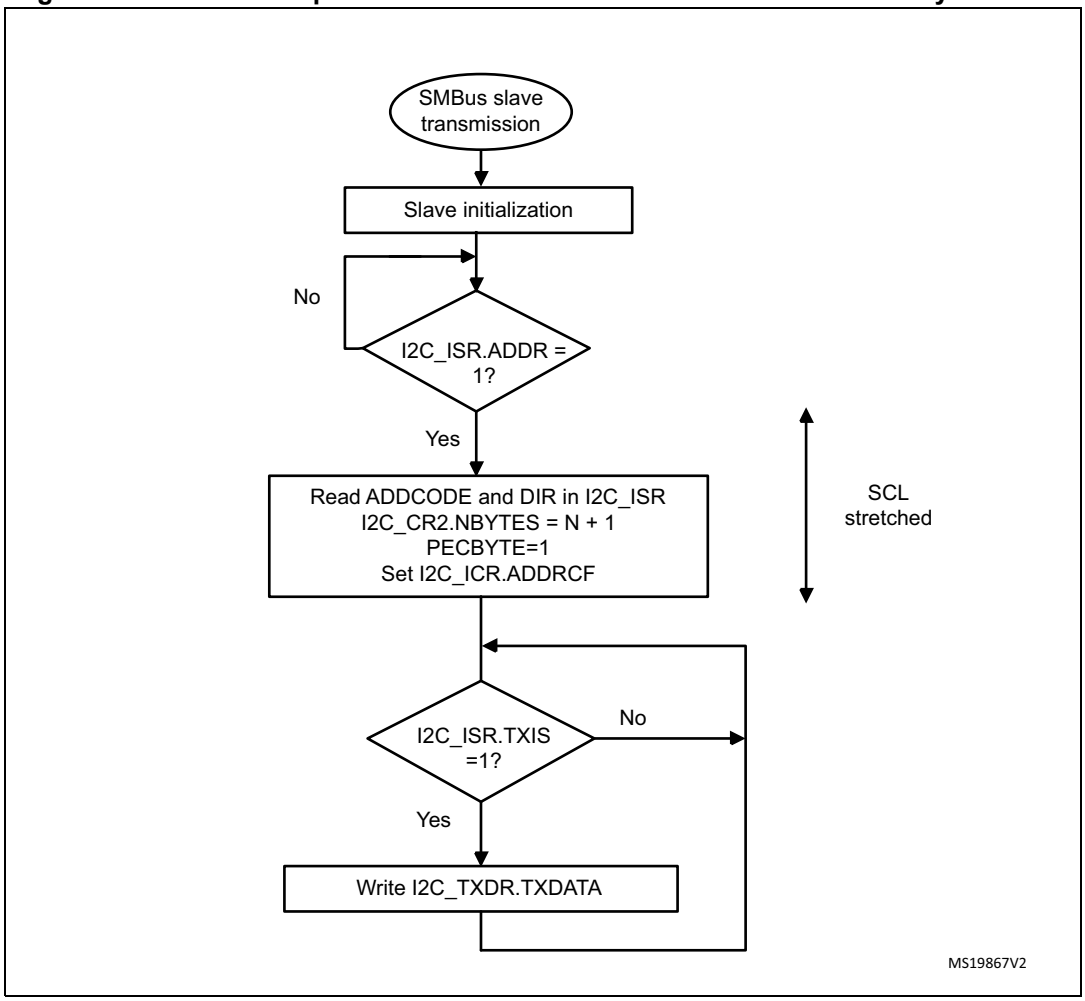
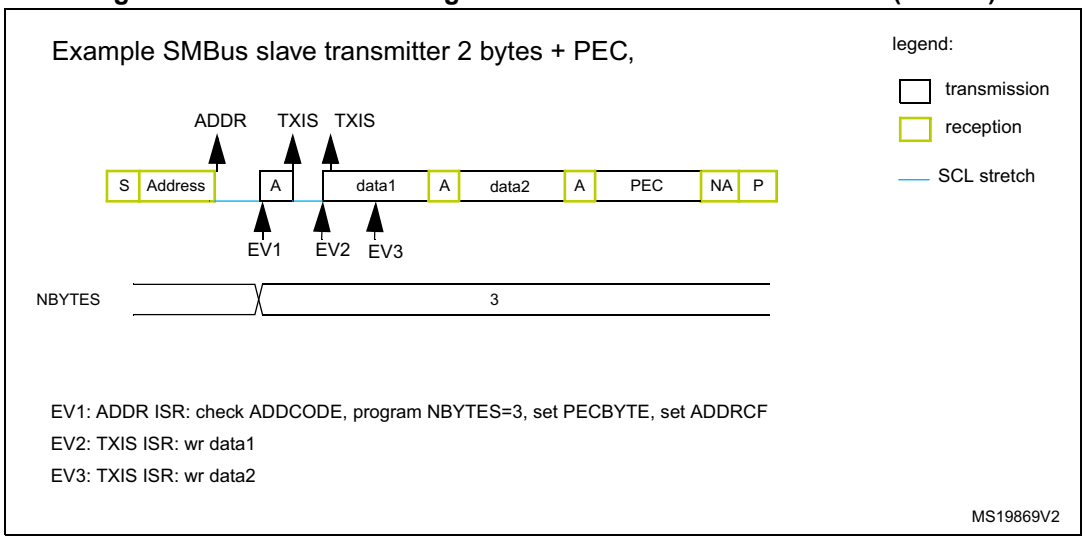


Figure 423. Transfer bus diagrams for SMBus slave transmitter (SBC=1)



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 1503* 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 424. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

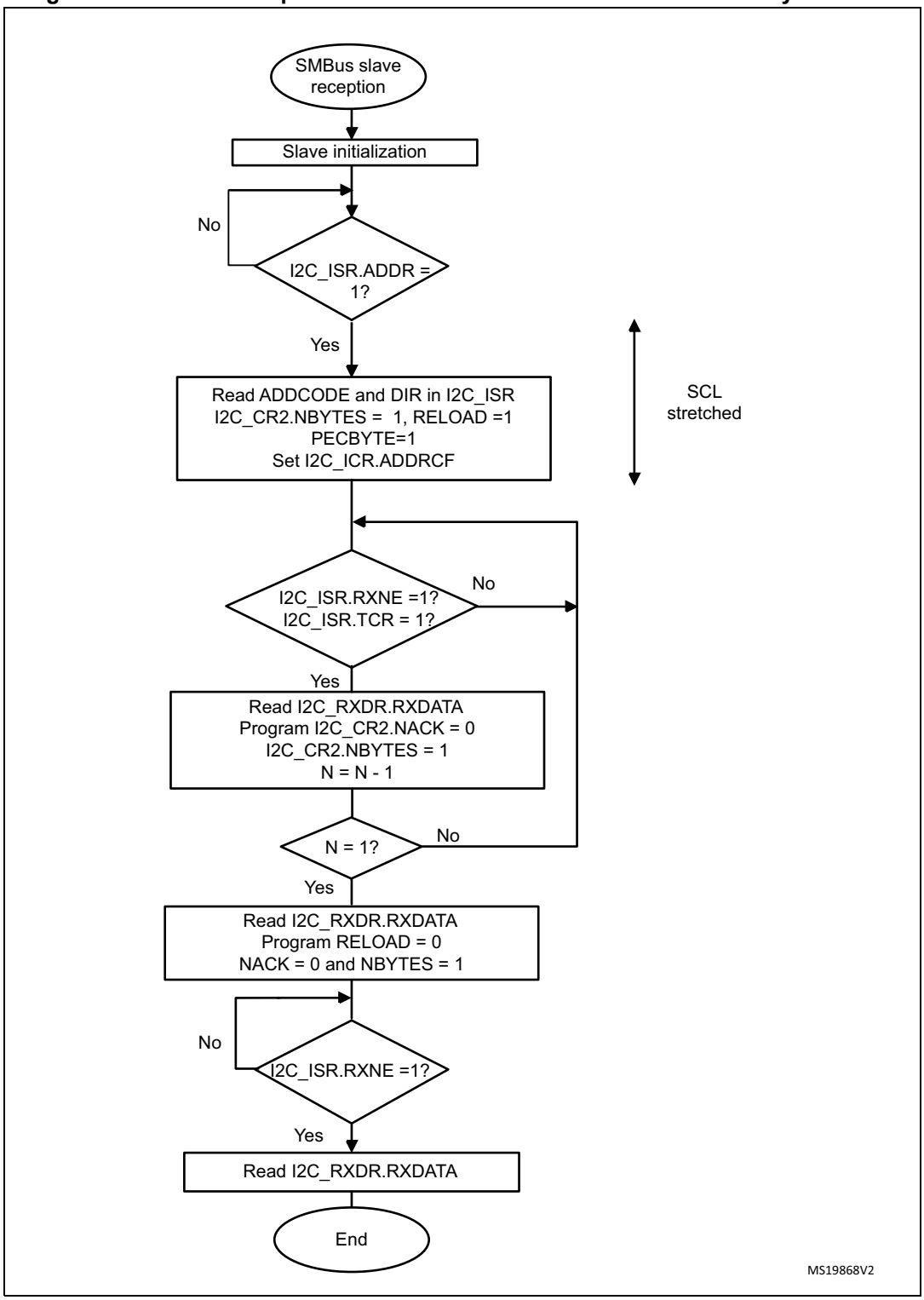
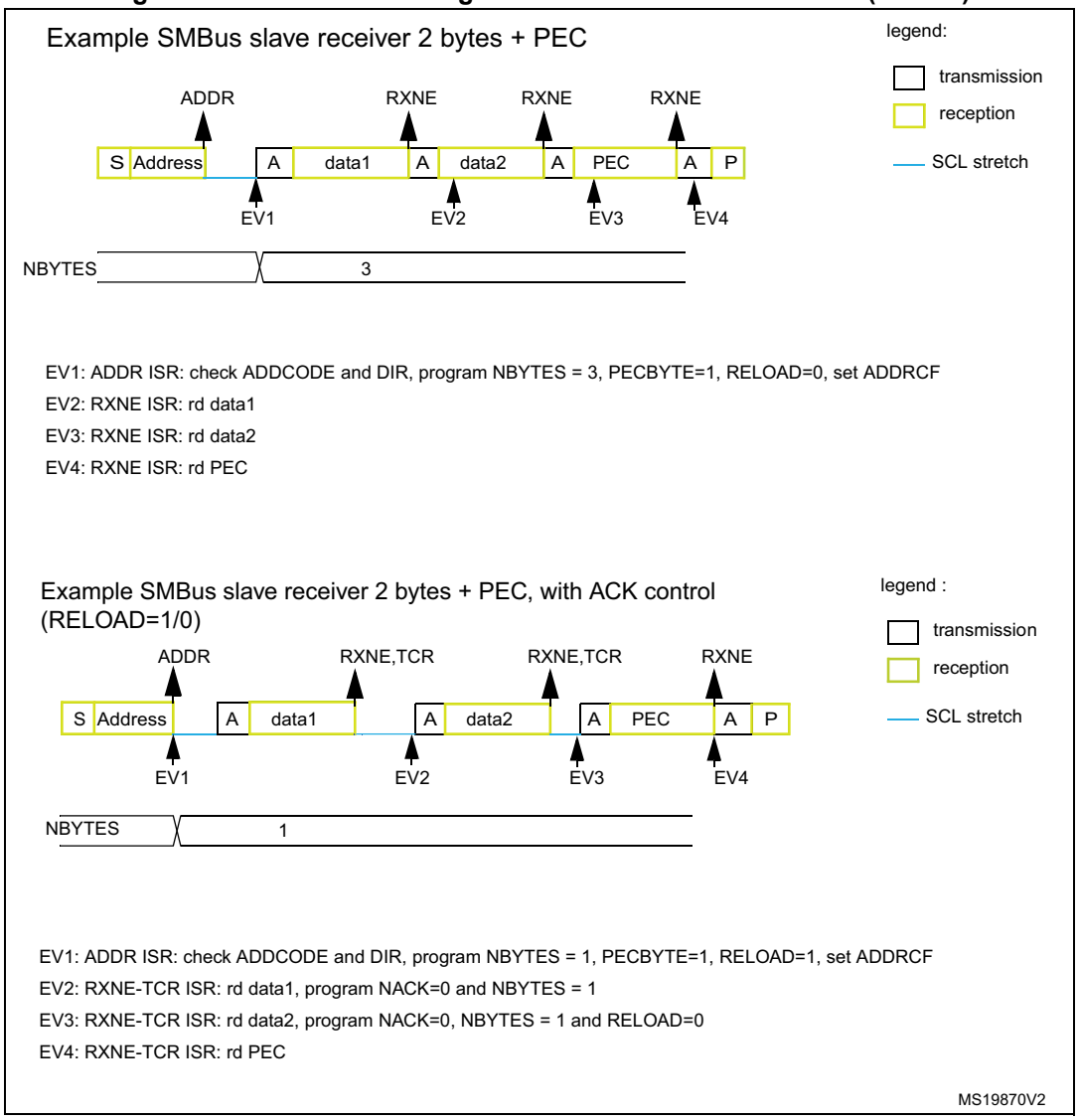


Figure 425. Bus transfer diagrams for SMBus slave receiver (SBC=1)



This section is relevant only when the SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

In addition to I2C master transfer management (refer to *Section 43.4.9: I2C master mode*) some additional software flowcharts are provided to support the SMBus.

SMBus master transmitter

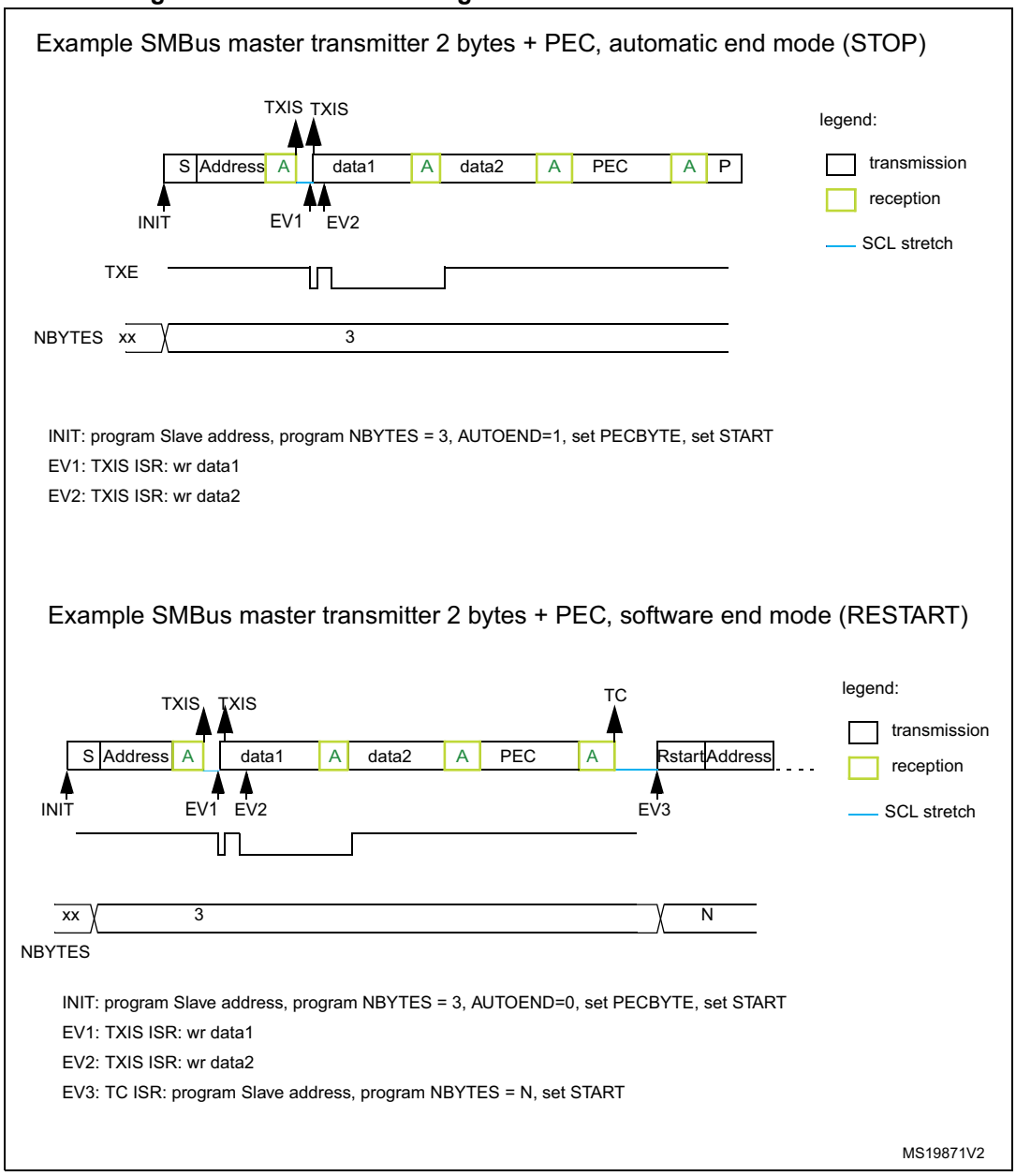
When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the 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 426. Bus transfer diagrams for SMBus master transmitter



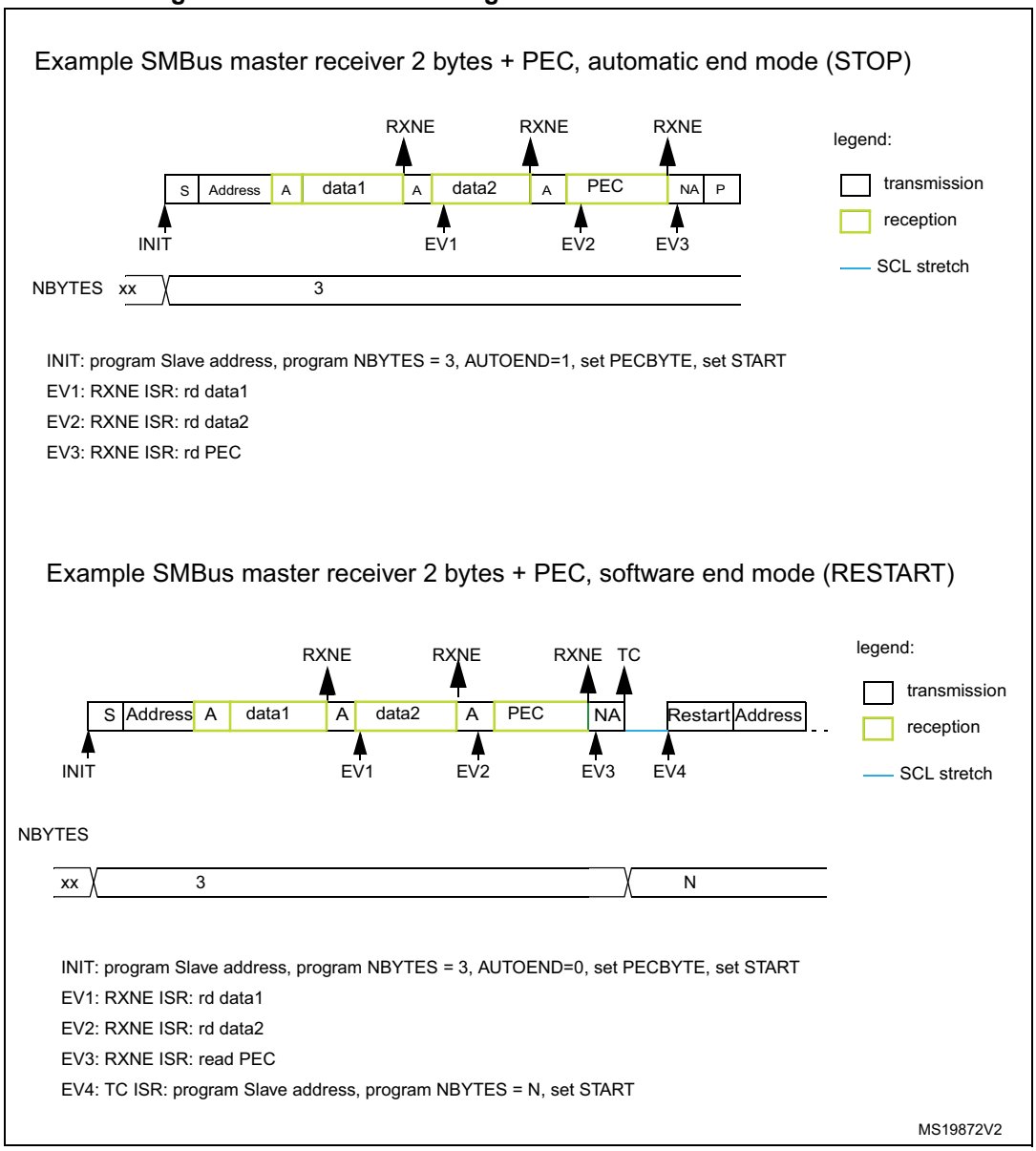
SMBus master receiver

When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 427. Bus transfer diagrams for SMBus master receiver



43.4.15 Wakeup from Stop mode on address match

This section is relevant only when wakeup from Stop mode feature is supported. Refer to *Section 43.3: 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).

43.4.16 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 43.3: 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 43.3: 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 43.3: 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.

43.4.17 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 1514*.
- 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 1529* and *SMBus master transmitter on page 1532*.

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 14: Direct memory access controller (DMA) on page 481*) 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 43.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to *SMBus Slave receiver on page 1530* and *SMBus master receiver on page 1534*.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

43.4.18 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_SMBUS_TIMEOUT configuration bits in the DBG module.

43.5 I2C low-power modes

Table 332. 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)(2)	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 43.3: 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.
2. Only I2C3 supports wakeup from Stop 2 mode. The other I2C instances must be disabled before entering Stop 2 mode.

43.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 333. I2C Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop 0, Stop 1, Stop 2 mode	Exit the Standby, Shutdown 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 43.3: I2C implementation*.

43.7 I2C registers

Refer to *Section 1.2 on page 76* for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

43.7.1 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 43.3: 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 43.3: 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 43.3: 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 43.3: 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 43.3: 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.

43.7.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	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 43.3: 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.

43.7.3 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.

43.7.4 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.

43.7.5 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 1495*) and for SCL high and low level counters (refer to *I2C master initialization on page 1510*).

$t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$

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}.

$t_{SCLDEL} = (SCLDEL+1) \times t_{PRESC}$

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}.

$t_{SDADEL} = SDADEL \times t_{PRESC}$

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.

$t_{SCLH} = (SCLH+1) \times t_{PRESC}$

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.

$t_{SCLL} = (SCLL+1) \times t_{PRESC}$

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.

43.7.6 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 43.3: I2C implementation.

43.7.7 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 43.3: 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 43.3: 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 43.3: 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.

43.7.8 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 43.3: 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 43.3: 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 43.3: 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.

43.7.9 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 43.3: I2C implementation.

43.7.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

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

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

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

Bits 7:0 **RXDATA[7:0]** 8-bit receive data
Data byte received from the I^2C bus

43.7.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

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

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

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

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data
Data byte to be transmitted to the I^2C bus
Note: These bits can be written only when TXE=1.

43.7.12 I2C hardware configuration register (I2C_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0111

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **WKP[3:0]**

0: Wakeup from Stop mode not implemented
1: Wakeup from Stop mode implemented

Bits 7:4 **ASYN[3:0]**

0: Independent kernel clock not implemented
1: Independent kernel clock implemented

Bits 3:0 **SMBUS[3:0]**

0: SMBus mode not implemented
1: SMBus mode implemented

43.7.13 I2C version register (I2C_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

43.7.14 I2C identification register (I2C_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0012

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

Bits 31:0 **ID[31:0]**: Identifier.

43.7.15 I2C size identification register (I2C_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

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

Bits 31:0 **SID[31:0]**: Size identifier.

43.7.16 I2C register map

The table below provides the I2C register map and reset values.

Table 334. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	**I2C_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x4	**I2C_CR2**	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	**I2C_OAR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0xC	**I2C_OAR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	**I2C_ TIMINGR**	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**I2C_ TIMEOUTR**	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**I2C_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	**I2C_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	**I2C_PECR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	**I2C_RXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 334. 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
0x03F0	**I2C_HWCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				WKP[3:0]				ASYN[3:0]				SMBUS[3:0]			
	Reset value																					0	0	0	1	0	0	0	1	0	0	0	1
0x03F4	**I2C_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																																
0x03F8	**I2C_IPIDR**	ID[31:16]																ID[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0x03FC	**I2C_SIDR**	SID[31:16]																SID[15:0]															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.3 on page 87* for the register boundary addresses.

44 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

44.1 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.

44.2 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

44.3 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

44.4 USART implementation

The table below describes USART implementation on STM32L552xx and STM32L562xx devices. It also includes LPUART for comparison.

Table 335. STM32L552xx and STM32L562xx features

USART / LPUART instances	STM32L552, STM32L562
USART1	FULL
USART2	FULL
USART3	FULL
UART4	BASIC
UART5	BASIC
LPUART1	LP

Table 336. USART / LPUART features

USART / LPUART modes/features(1)	Full feature set	Basic feature set	Low-power feature set
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

Table 336. USART / LPUART features (continued)

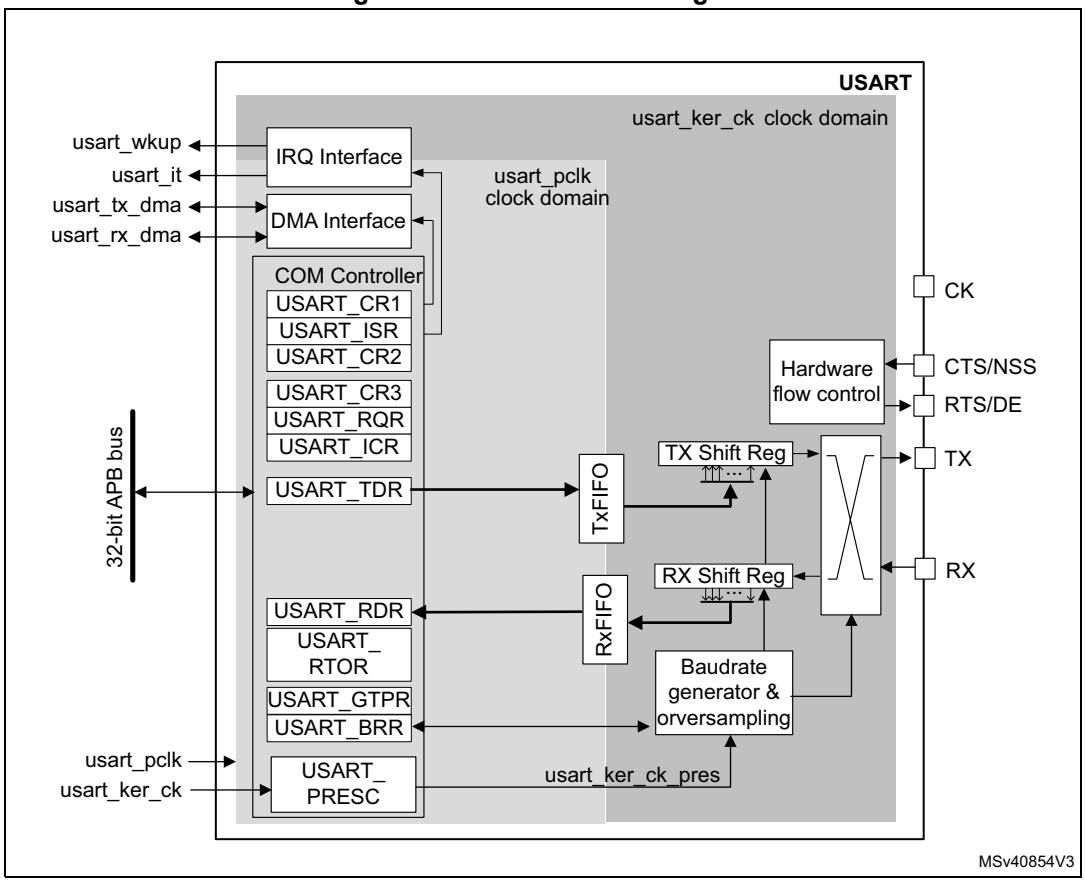
USART / LPUART modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver Enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size	8		

1. X = supported.

44.5 USART functional description

44.5.1 USART block diagram

Figure 428. USART block diagram



The simplified block diagram given in *Figure 428* 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.

44.5.2 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.

44.5.3 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 429*):

- 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.

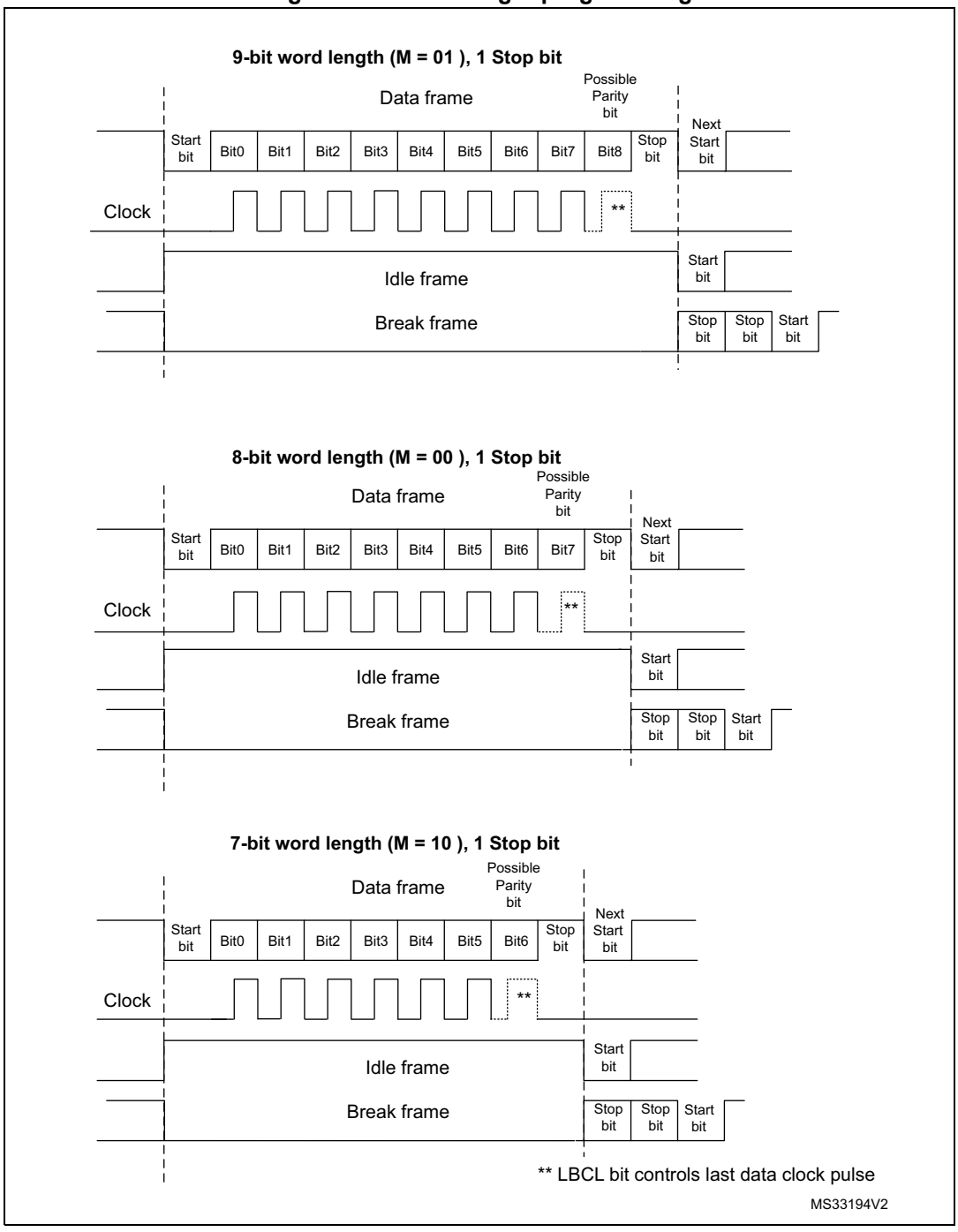
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

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 429. Word length programming



44.5.4 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.

44.5.5 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

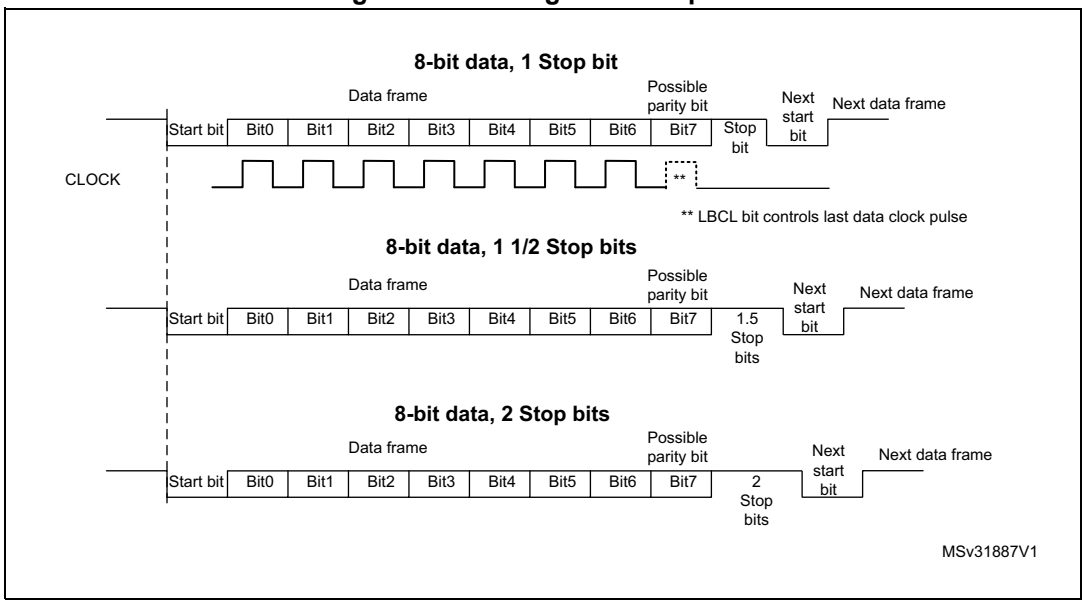
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***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 430*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 430. 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 44.5.10: 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.

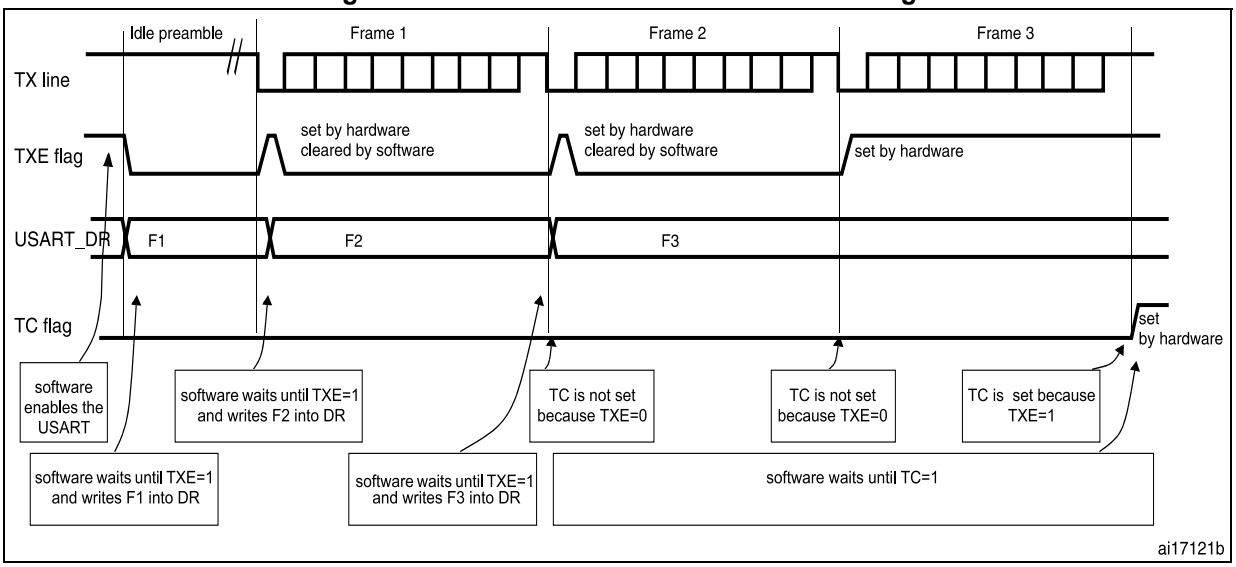
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  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 431: TC/TXE behavior when transmitting*).

Figure 431. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 429*).

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.

44.5.6 USART receiver

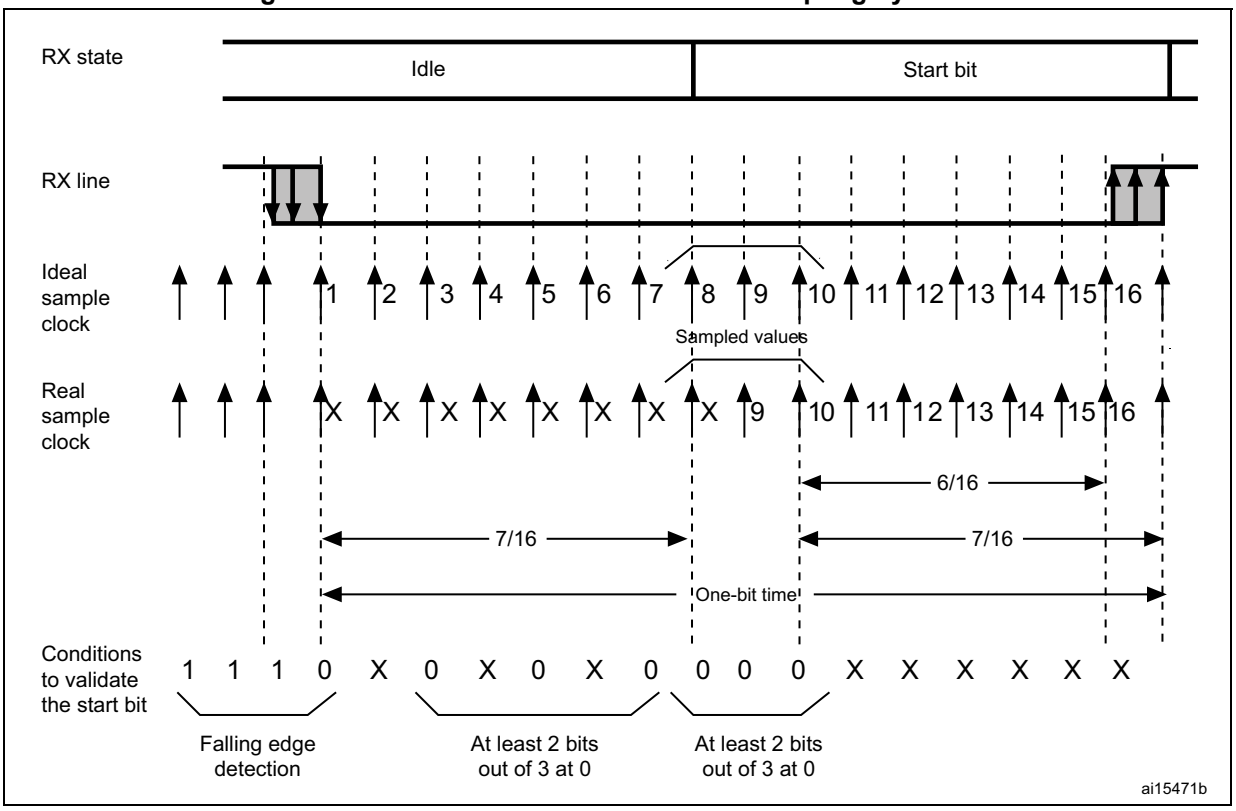
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

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 432. Start bit detection when oversampling by 16 or 8



Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at ‘0’ (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at ‘0’ and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at ‘0’).

The start bit is validated but the NE noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at ‘0’ (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at ‘0’.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 44.5.10: 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:

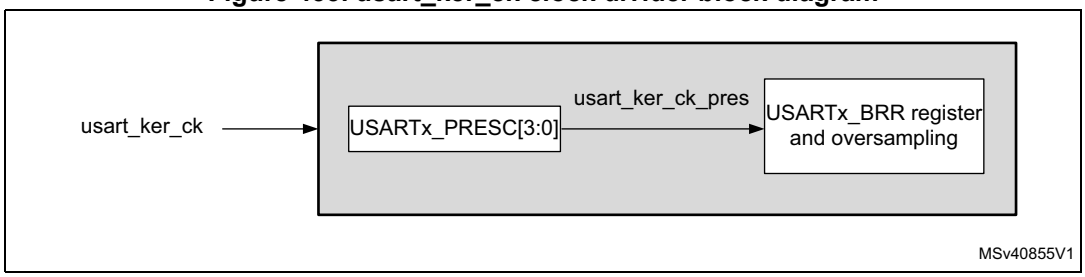
- Possible use of the USART in low-power mode
- Communication speed.

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 433. 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 434* and *Figure 435*).

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 44.5.8: Tolerance of the USART receiver to clock deviation on page 1578*)
- 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 337*) 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 44.5.8: Tolerance of the USART receiver to clock deviation on page 1578*). 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 434. Data sampling when oversampling by 16

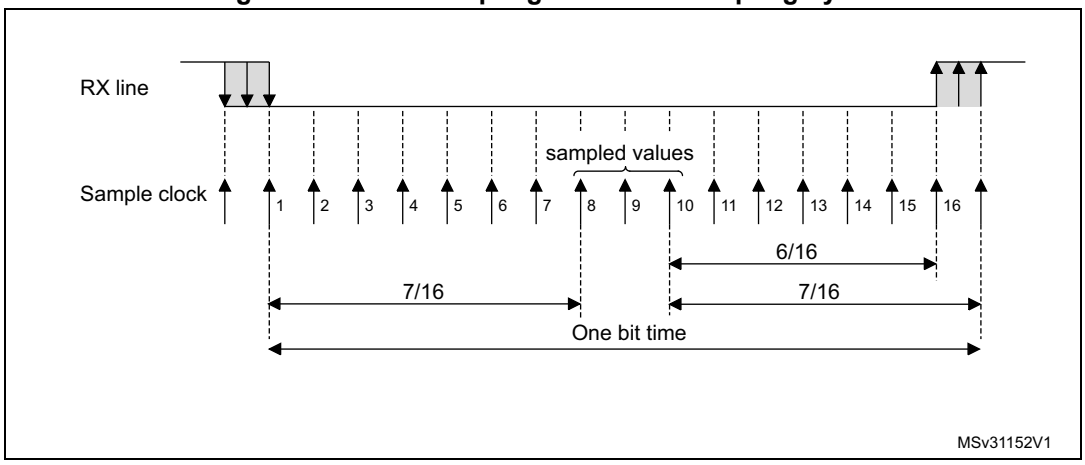


Figure 435. Data sampling when oversampling by 8

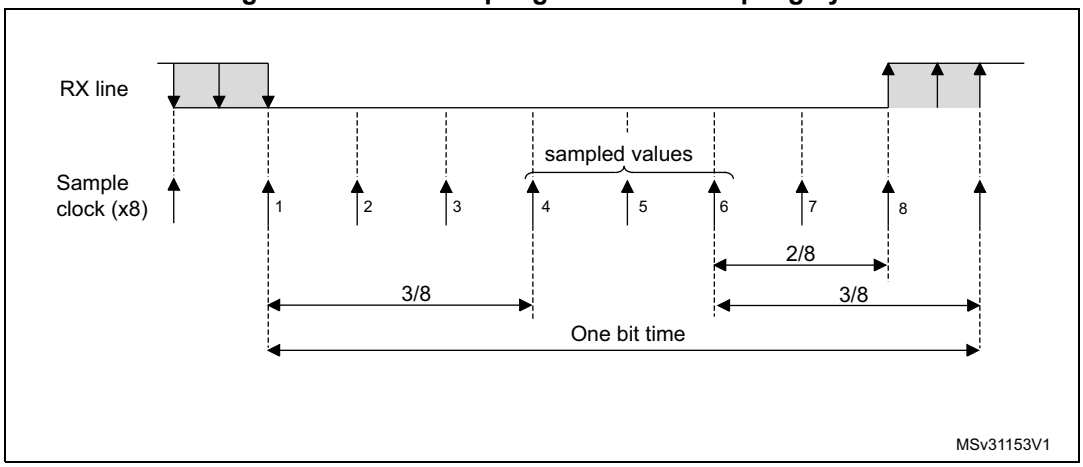


Table 337. Noise detection from sampled data

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 44.5.16: USART receiver timeout on page 1592* 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.

44.5.7 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

44.5.8 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 338*, *Table 339*, 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 338. 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 339. 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 338 and Table 339 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).

44.5.9 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.

44.5.10 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:

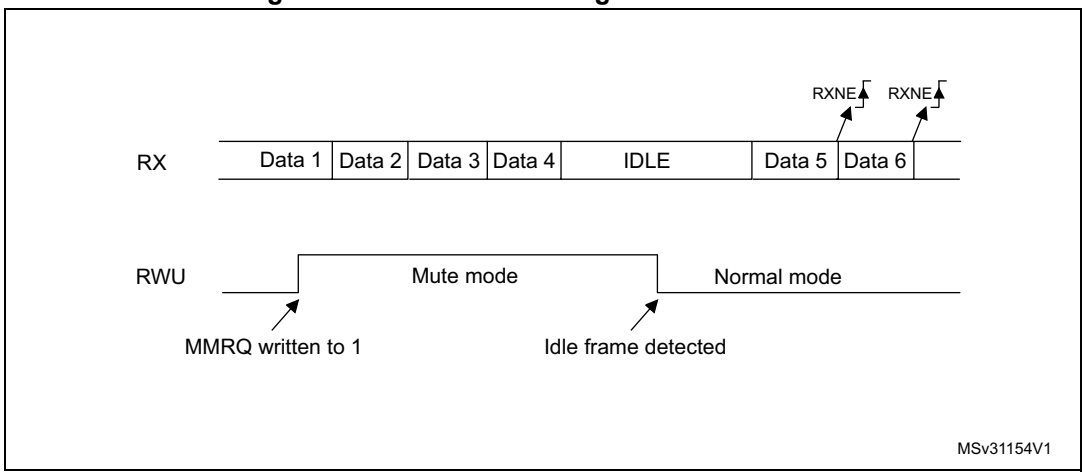
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

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 436*.

Figure 436. 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.

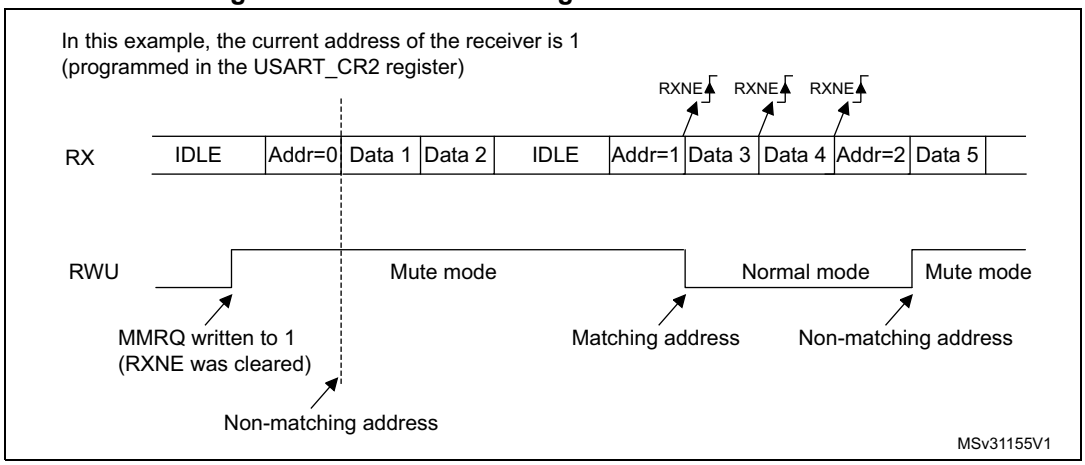
The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 437*.

Figure 437. Mute mode using address mark detection



44.5.11 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.

44.5.12 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 340*.

Table 340. 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).

44.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 44.4: USART implementation on page 1561*.

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 44.5.4* 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 438: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 1587*.

Examples of break frames are given on *Figure 439: Break detection in LIN mode vs. Framing error detection on page 1588*.

Figure 438. Break detection in LIN mode (11-bit break length - LBDL bit is set)

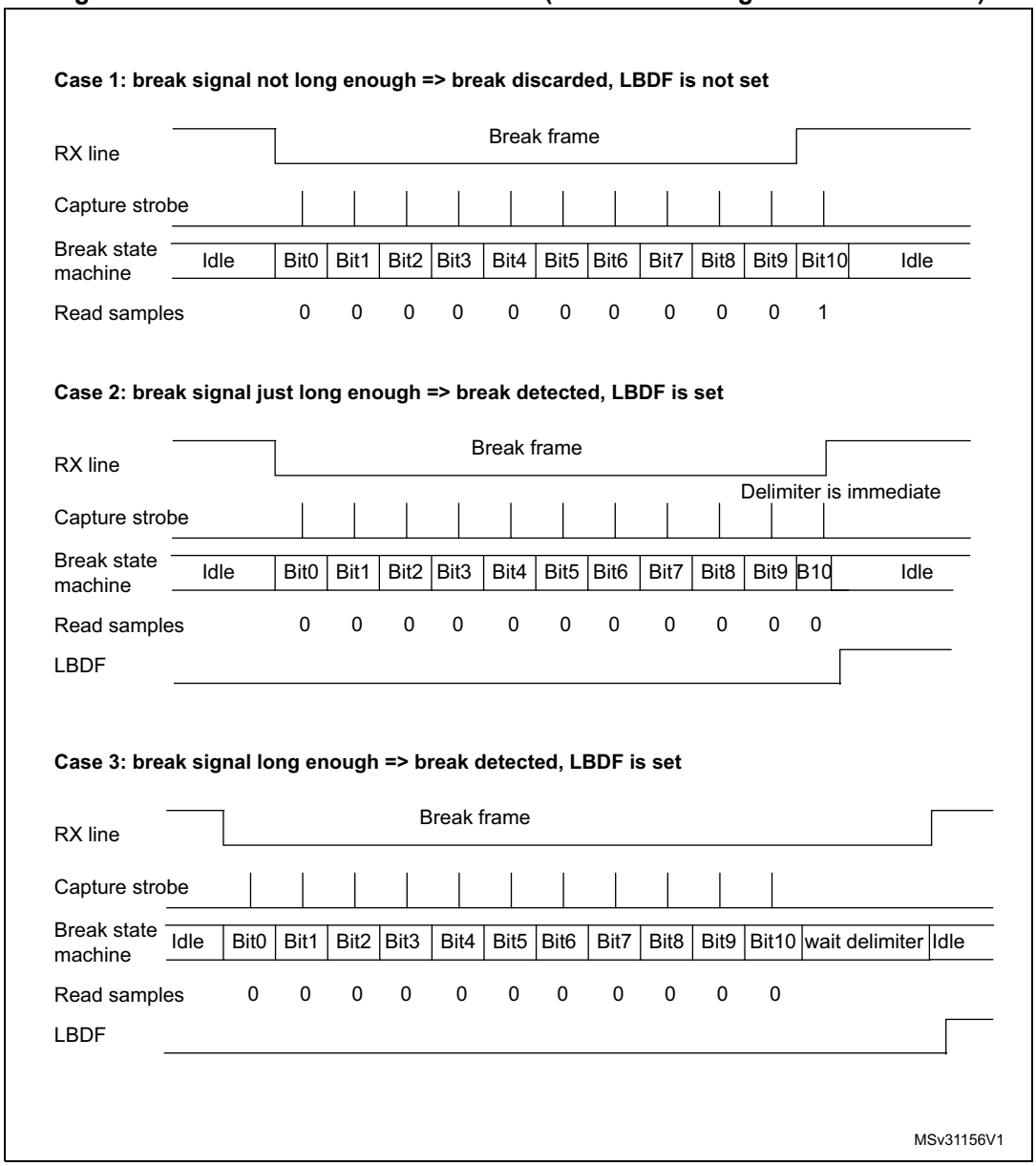
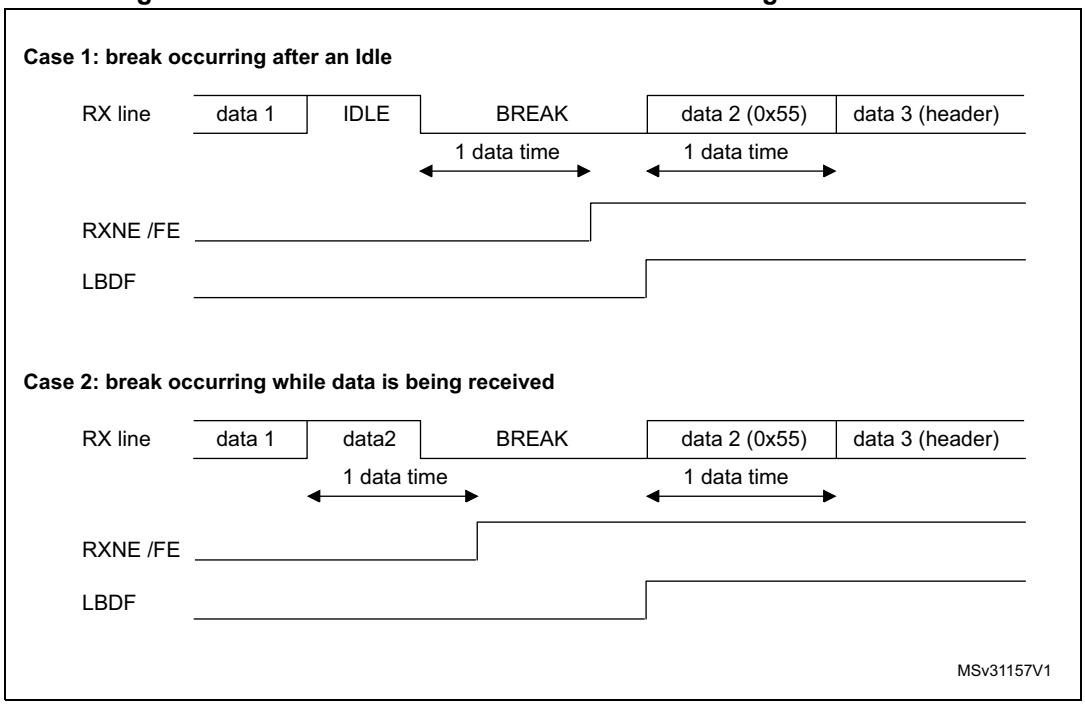


Figure 439. Break detection in LIN mode vs. Framing error detection



44.5.14 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 440*, *Figure 441* and *Figure 442*).

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 440. USART example of synchronous master transmission

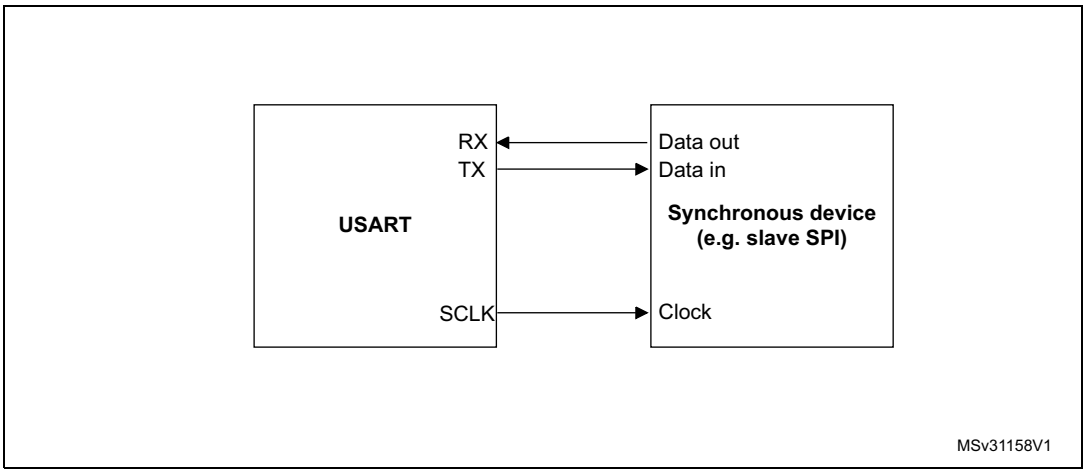


Figure 441. USART data clock timing diagram in synchronous master mode (M bits = 00)

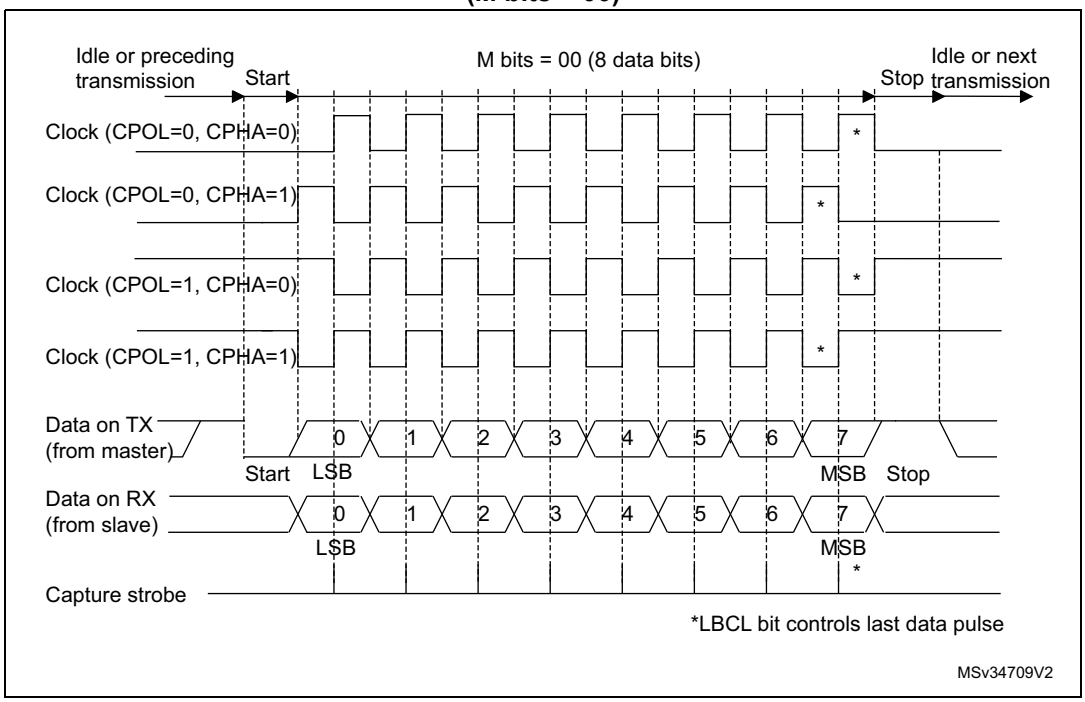
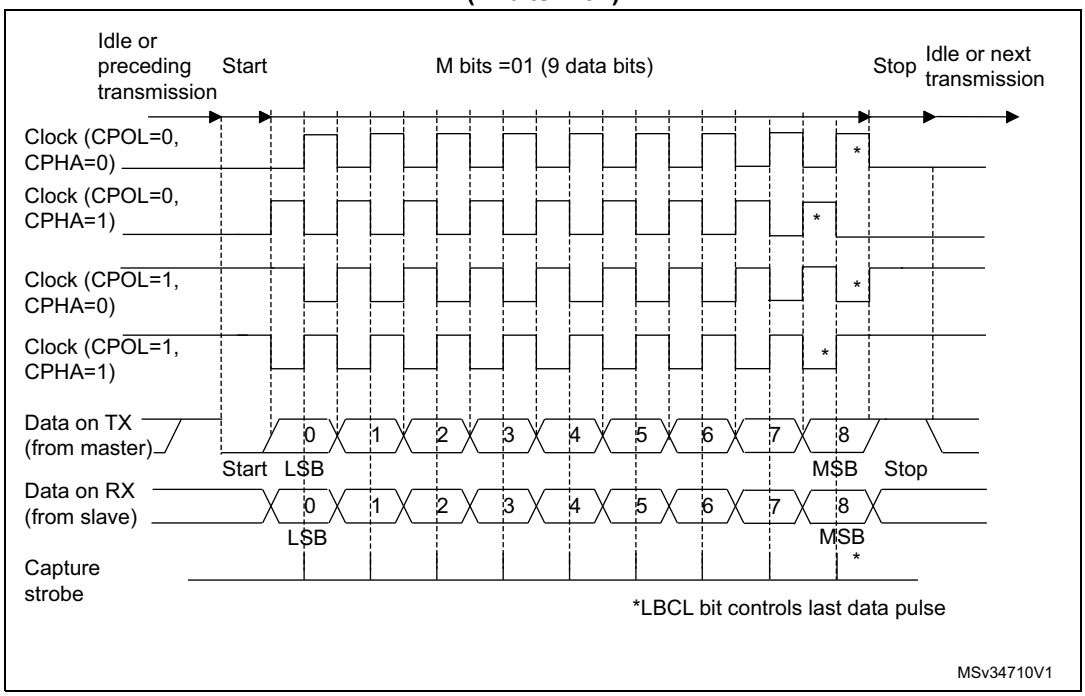


Figure 442. USART data clock timing diagram in synchronous master mode (M bits = 01)



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.

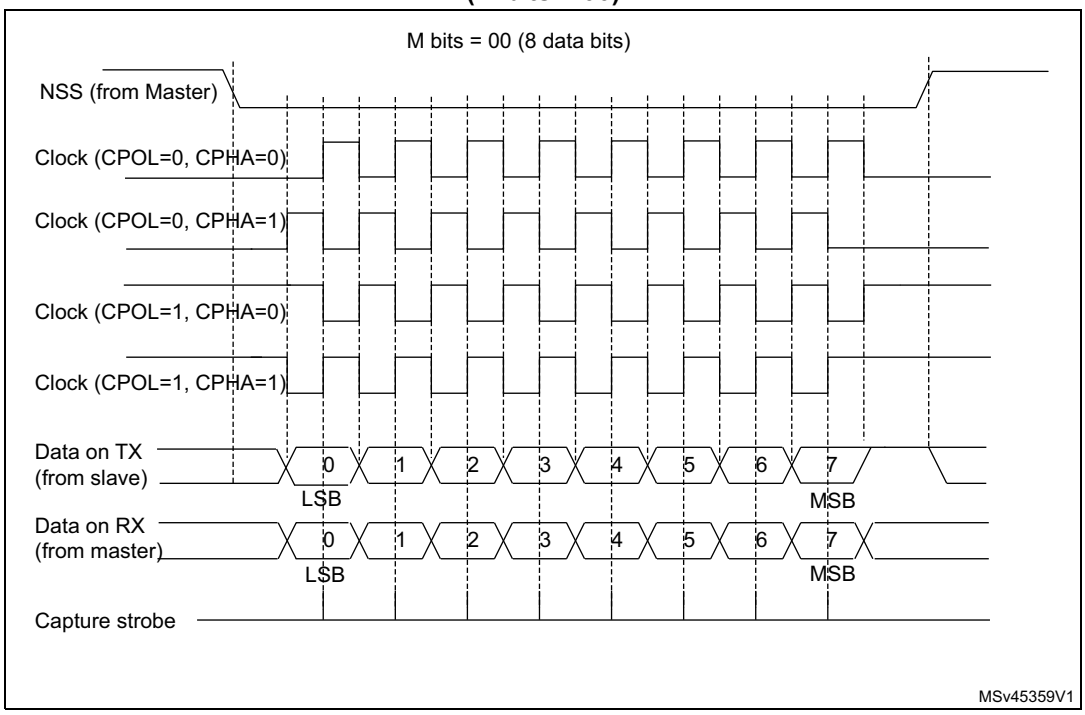
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

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 443*).

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 443. 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.*

44.5.15 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.

44.5.16 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.

44.5.17 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to *Section 44.4: USART implementation on page 1561*.

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.

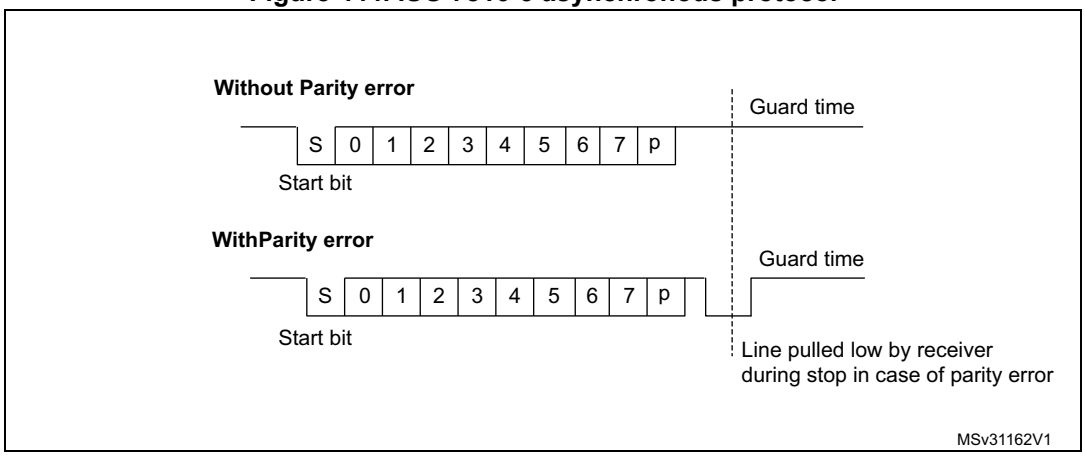
The USART should be configured as:

- 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 444 shows examples of what can be seen on the data line with and without parity error.

Figure 444. 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.

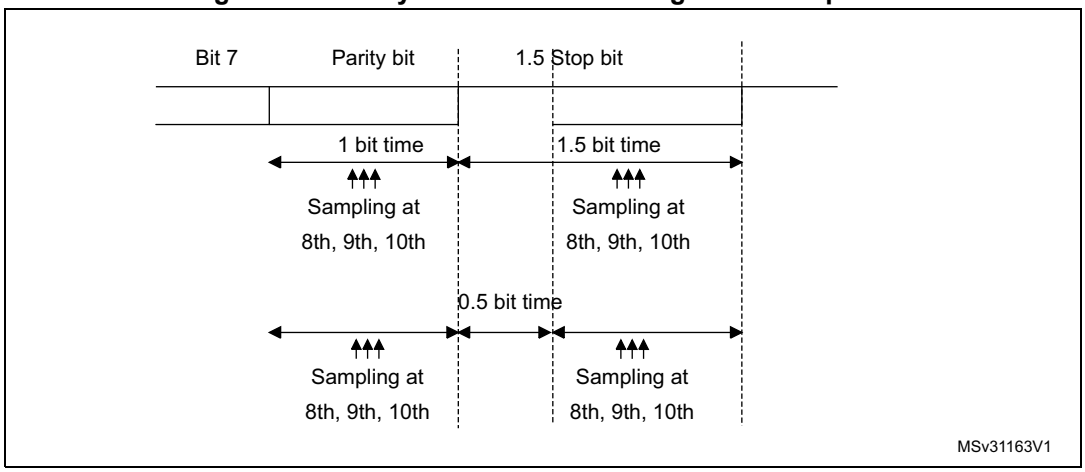
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it 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 445 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 445. 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.

44.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 44.4: USART implementation on page 1561*.

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 446*).

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 447*).
- 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 446. IrDA SIR ENDEC block diagram

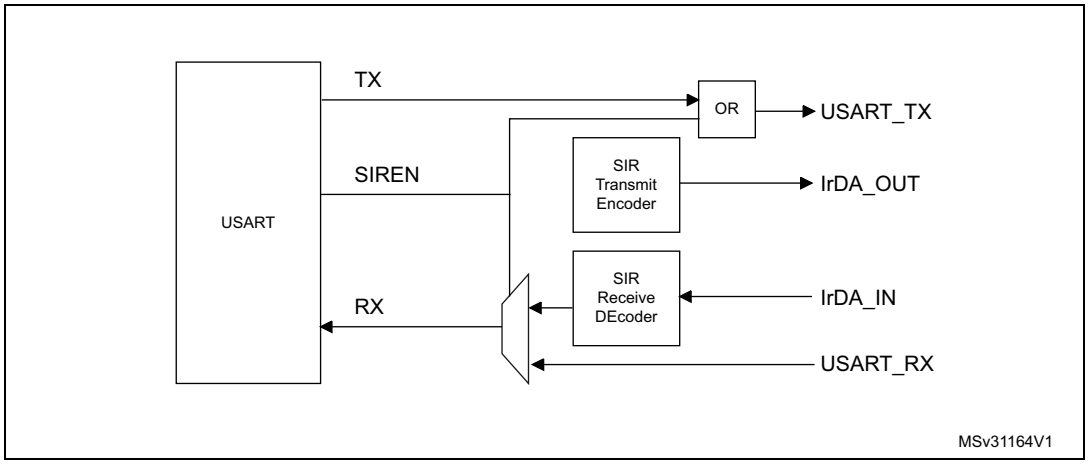
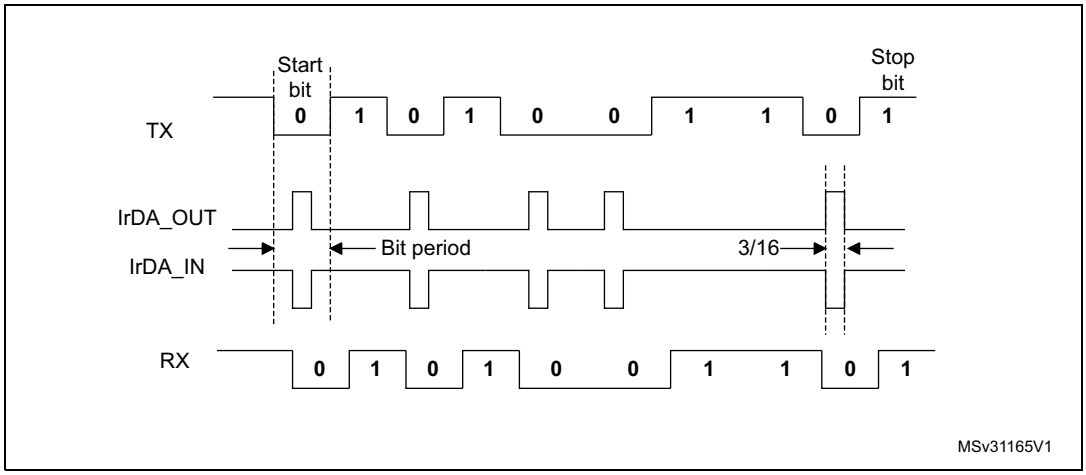


Figure 447. IrDA data modulation (3/16) - Normal mode



44.5.19 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 44.4: USART implementation on page 1561 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 44.5.6. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

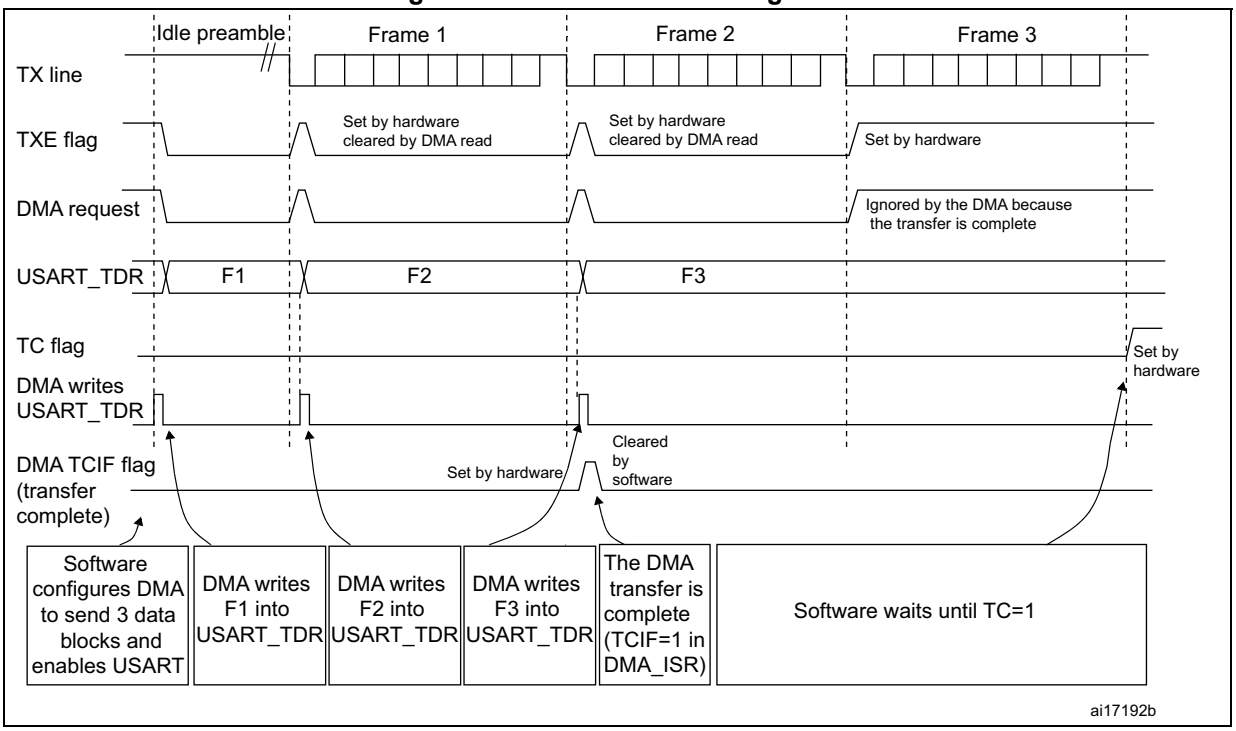
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode 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 448. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

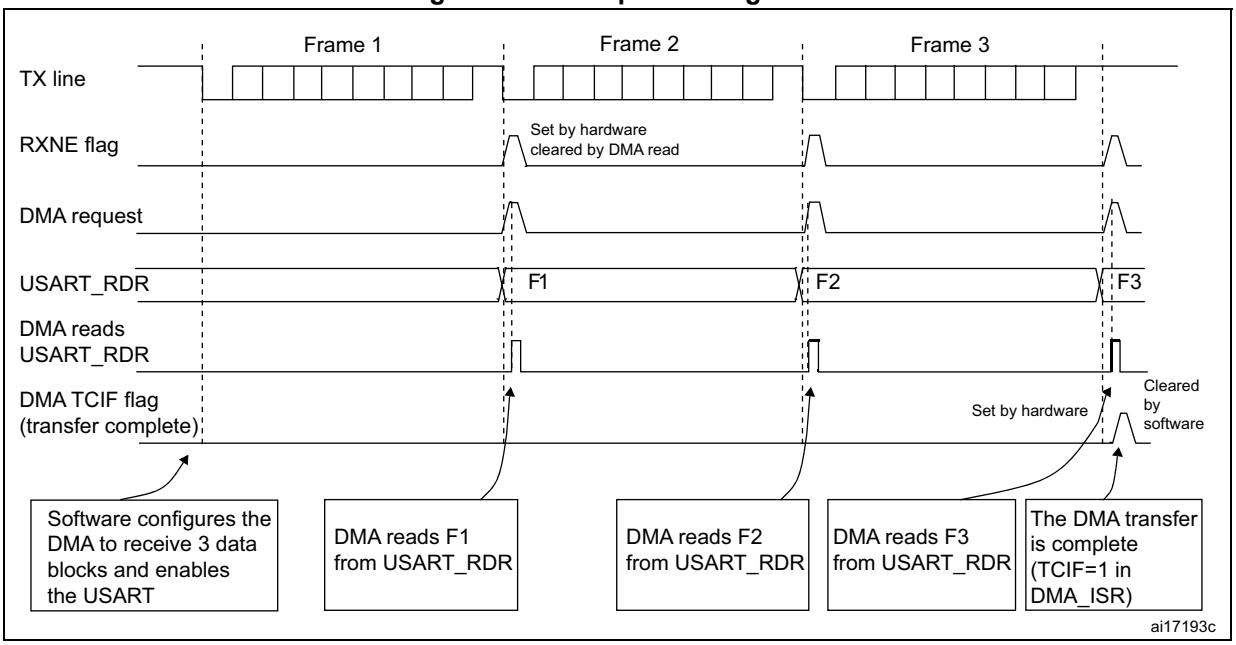
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 449. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

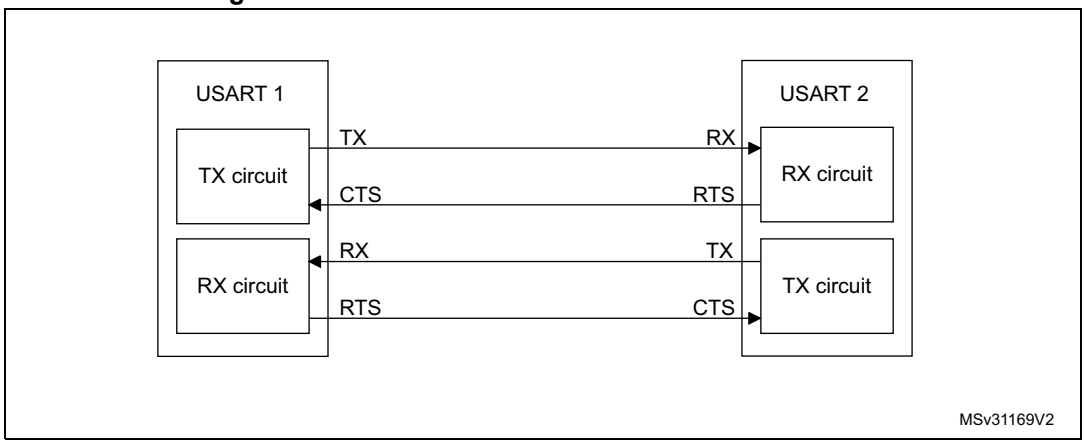
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

44.5.20 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 450* shows how to connect 2 devices in this mode:

Figure 450. Hardware flow control between 2 USARTs

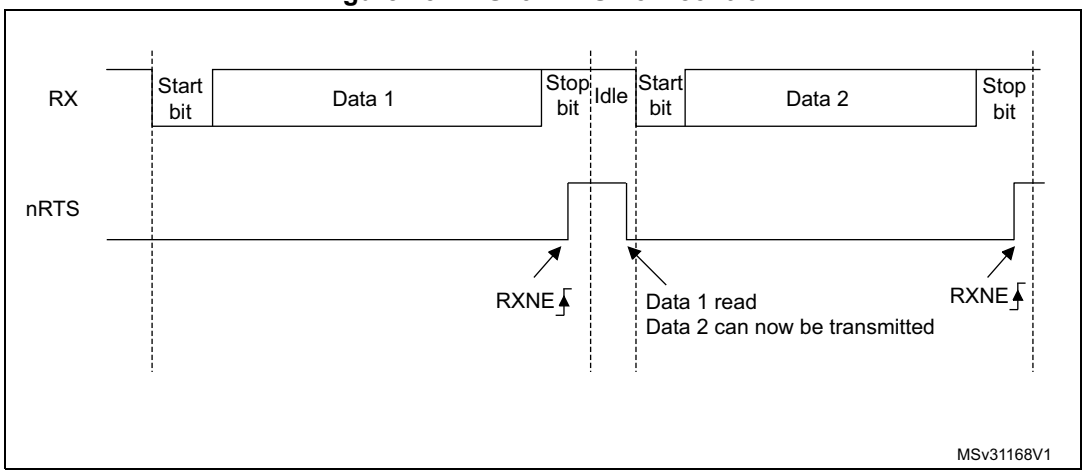


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to ‘1’ in the USART_CR3 register.

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 451* shows an example of communication with RTS flow control enabled.

Figure 451. RS232 RTS flow control



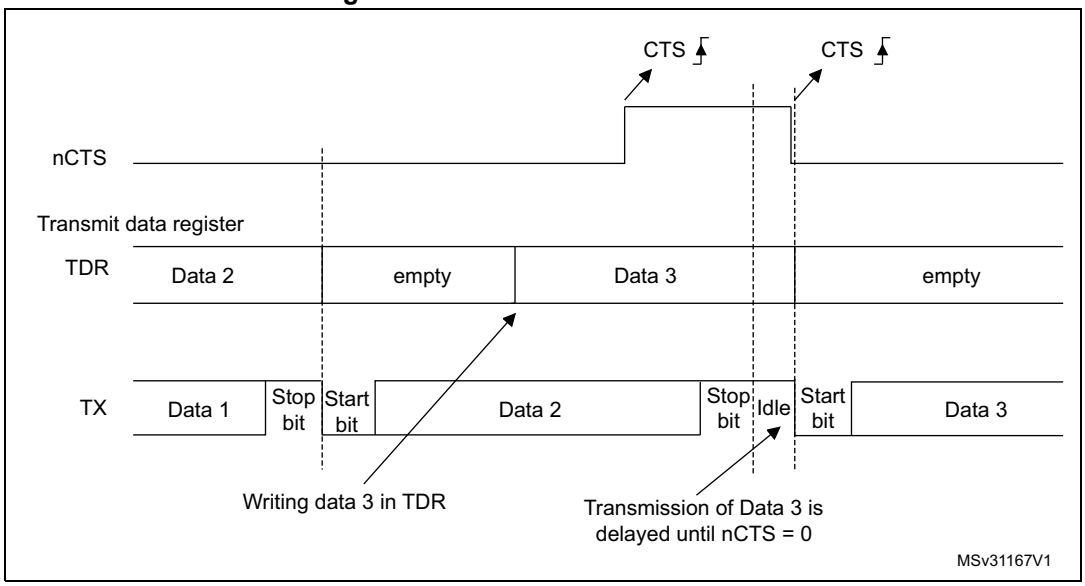
Note: When FIFO mode is enabled, nRTS is deasserted only when RXFIFO is full.

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 452* shows an example of communication with CTS flow control enabled.

Figure 452. 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).

44.5.21 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 453 shows the USART behavior when the wakeup event is verified.

Figure 453. Wakeup event verified (wakeup event = address match, FIFO disabled)

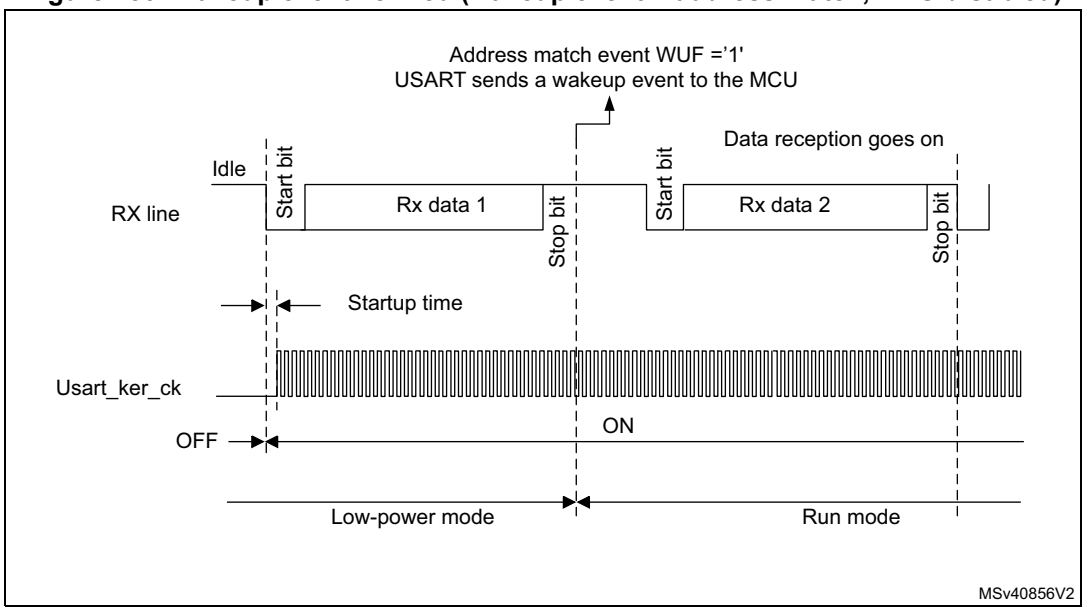
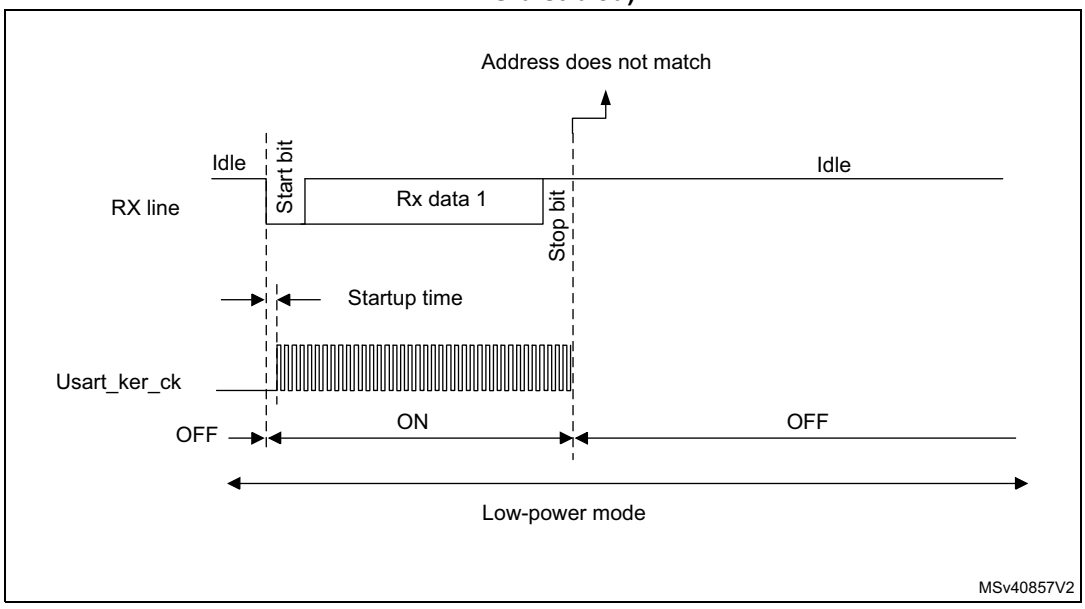


Figure 454 shows the USART behavior when the wakeup event is not verified.

Figure 454. 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 44.5.8: 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 338: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the usart_ker_ck inaccuracy.

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% - 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wakeup correctly from low-power mode is: 1/11.32 µs = 88.36 Kbaud.

44.6 USART in low-power modes

Table 341. 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 44.4: 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.

44.7 USART interrupts

Refer to *Table 342* for a detailed description of all USART interrupt requests.

Table 342. 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 44.4: 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.

44.8 USART registers

Refer to *Section 1.2 on page 76* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

44.8.1 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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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.

44.8.2 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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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.

44.8.3 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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 434* and *Figure 435*)

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

44.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS[1:0]		SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Remaining combinations: Reserved

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Remaining combinations: Reserved

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register
Note: If the USART does not support the Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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.

44.8.5 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.

44.8.6 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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

44.8.7 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 44.4: USART implementation on page 1561.

44.8.8 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 44.4: USART implementation on page 1561.

44.8.9 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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.5.8: Tolerance of the USART receiver to clock deviation on page 1578).

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.

44.8.10 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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.5.8: Tolerance of the USART receiver to clock deviation on page 1578).

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

44.8.11 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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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 44.4: USART implementation on page 1561.

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.

44.8.12 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 428*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

44.8.13 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 428*).

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.

44.8.14 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.

44.8.15 USART register map

The table below gives the USART register map and reset values.

Table 343. 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 343. 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 2.3: Memory organization* for the register boundary addresses.

45 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

45.1 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.

45.2 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

45.3 LPUART implementation

Below the description of LPUART implementation in comparison with USART.

Table 344. STM32L552xx and STM32L562xx features

USART / LPUART instances	STM32L552, STM32L562
USART1	FULL
USART2	FULL
USART3	FULL
UART4	BASIC
UART5	BASIC
LPUART1	LP

Table 345. USART / LPUART features

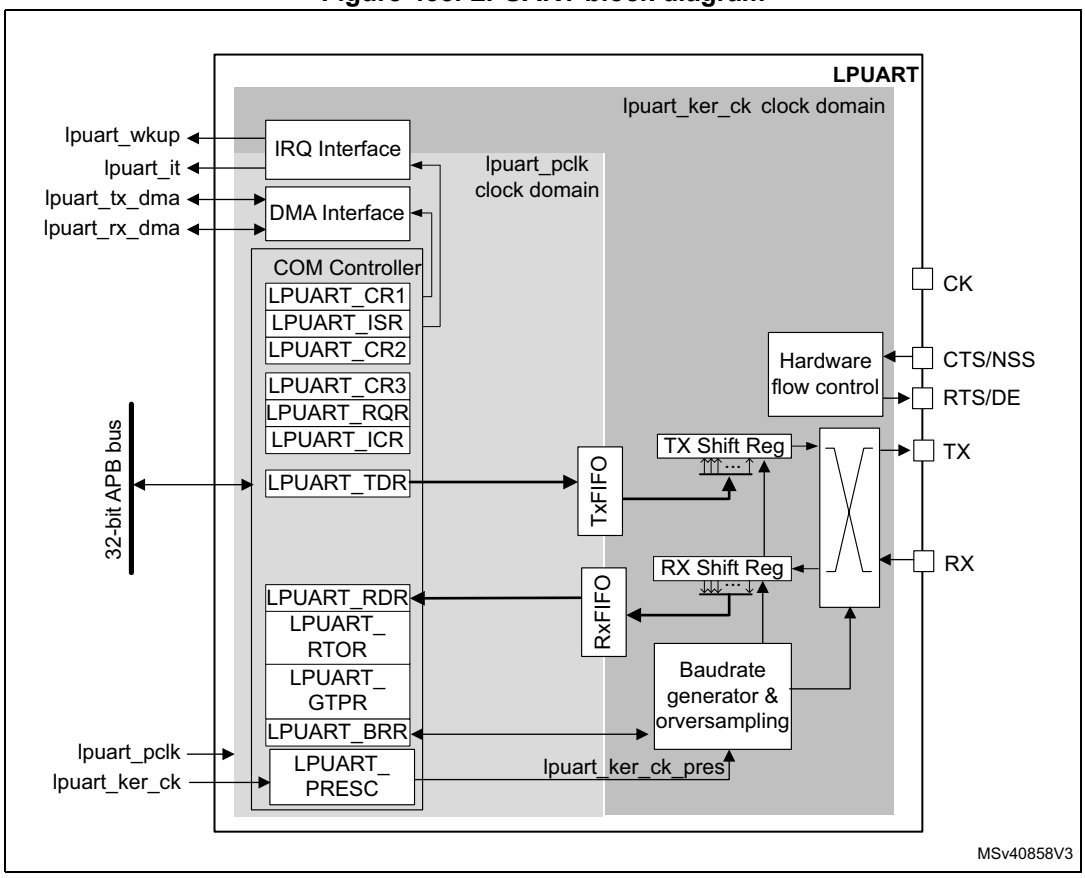
USART / LPUART modes/features(1)	USART1/2/3	UART4/5	LPUART1
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	-	-
Smartcard mode	X	-	-
Single-wire Half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
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.

45.4 LPUART functional description

45.4.1 LPUART block diagram

Figure 455. LPUART block diagram



The simplified block diagram given in *Figure 455* 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.

45.4.2 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.

45.4.3 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 429*).

- 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.

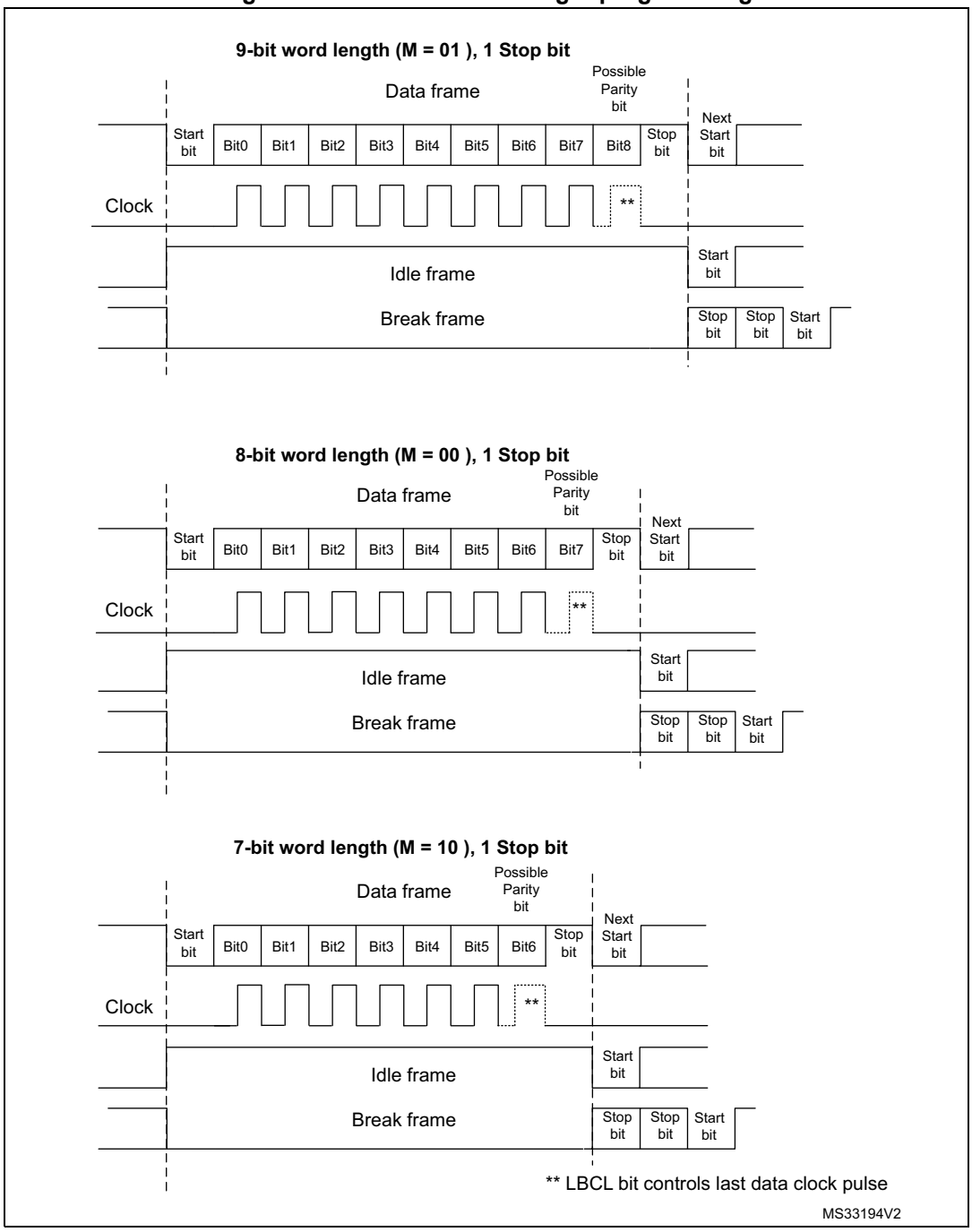
An ***Idle character*** is interpreted as an entire frame of “1”s. (The number of “1” ‘s includes the number of stop bits).

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 456. LPUART word length programming



45.4.4 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 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bits 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.

45.4.5 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 455*).

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

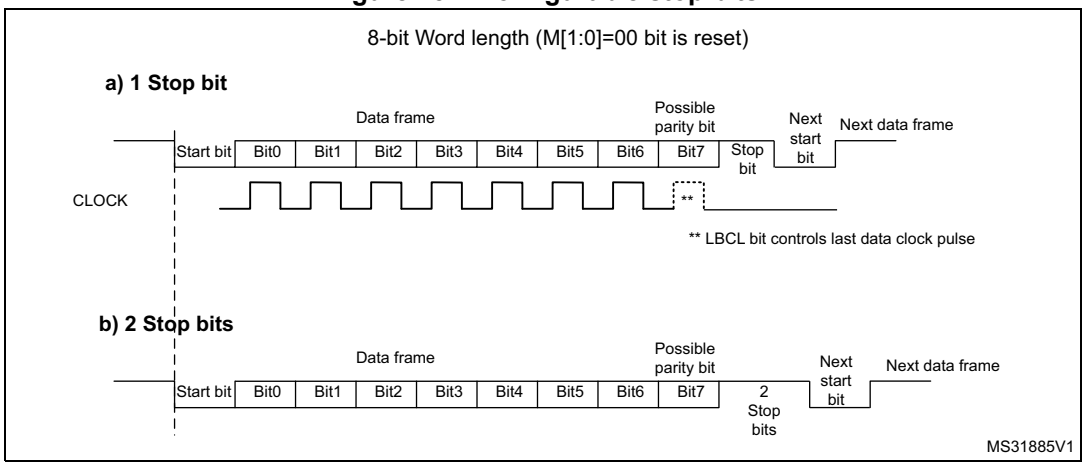
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13,12).

- ***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 457. 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 44.5.10: 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.

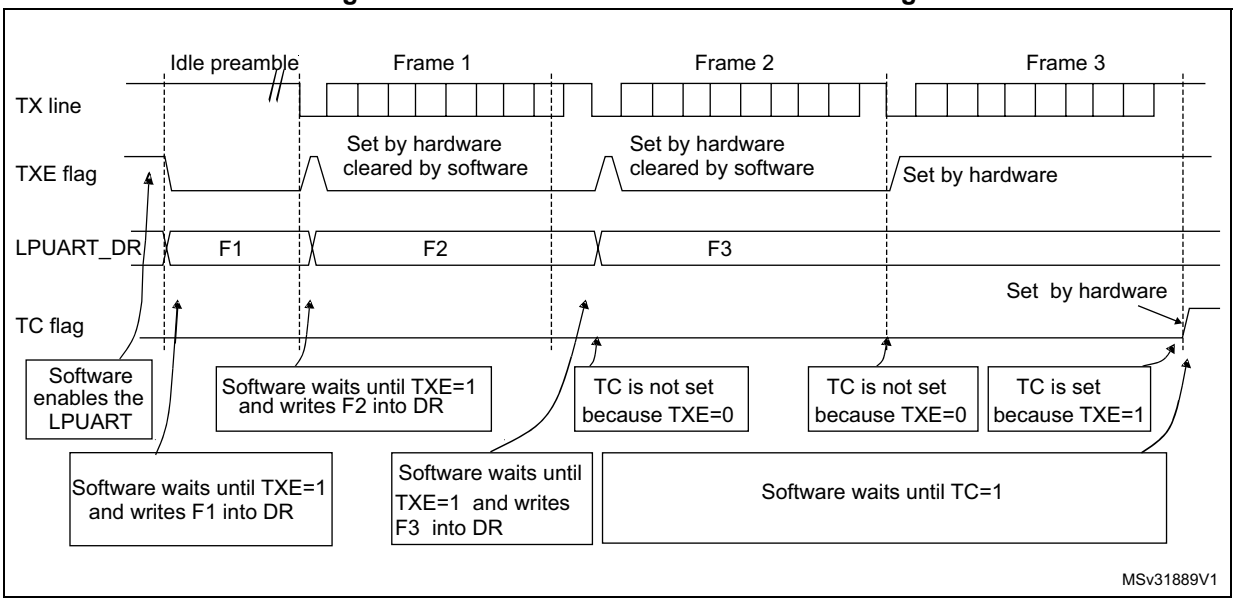
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

  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 458: TC/TXE behavior when transmitting*).

Figure 458. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 456*).

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.

45.4.6 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 44.5.10: 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:

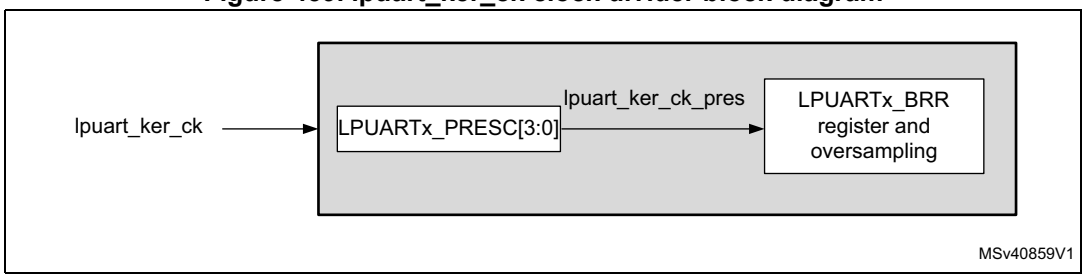
- Possible use of the LPUART in low-power mode
- Communication speed.

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 459. 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.

45.4.7 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 346. 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 347. 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

45.4.8 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 348*:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 348. 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 348 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’).

45.4.9 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:

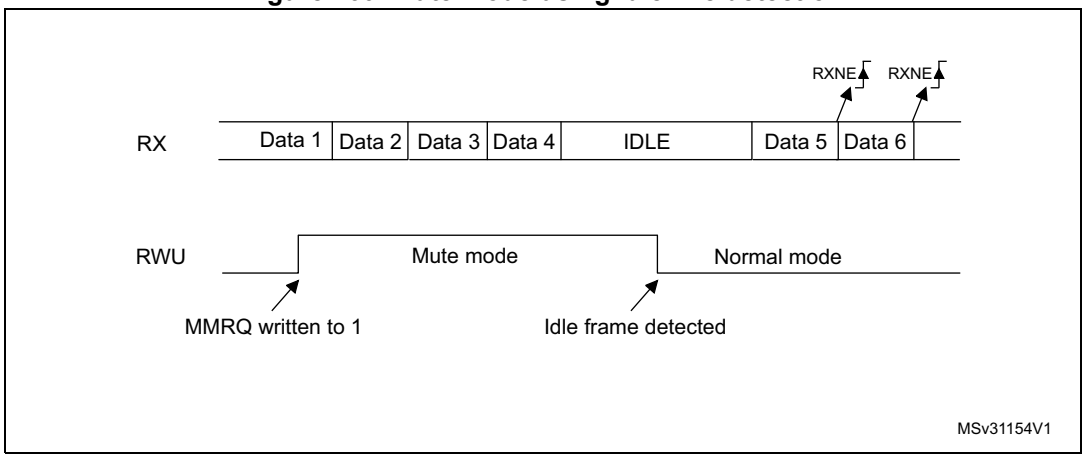
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

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 460*.

Figure 460. 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.

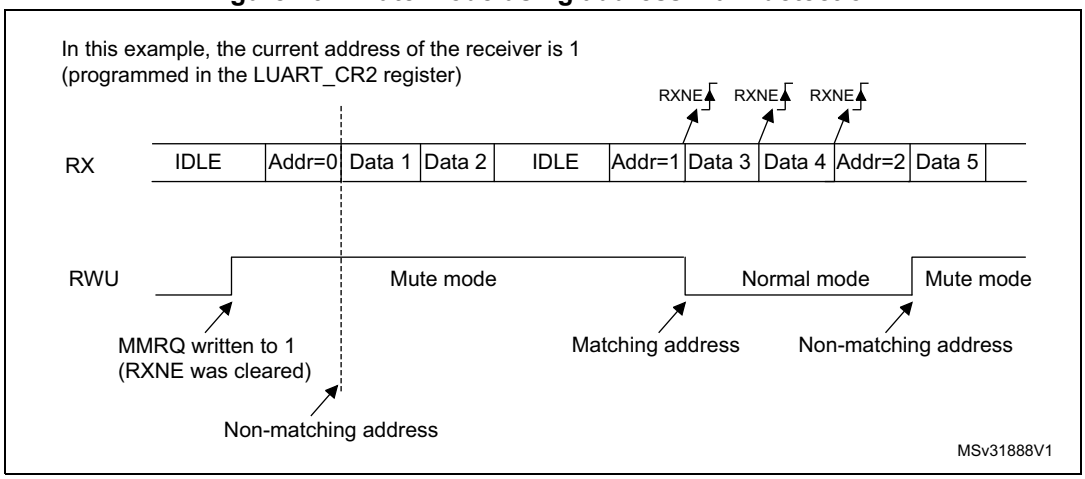
The LPUART also enters Mute mode when the MMRQ bit is written to ‘1’. The RWU bit is also automatically set in this case.

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 461*.

Figure 461. Mute mode using address mark detection



45.4.10 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 349*.

Table 349: 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)).

45.4.11 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.

45.4.12 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 44.4: USART implementation on page 1561 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 44.5.6. To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

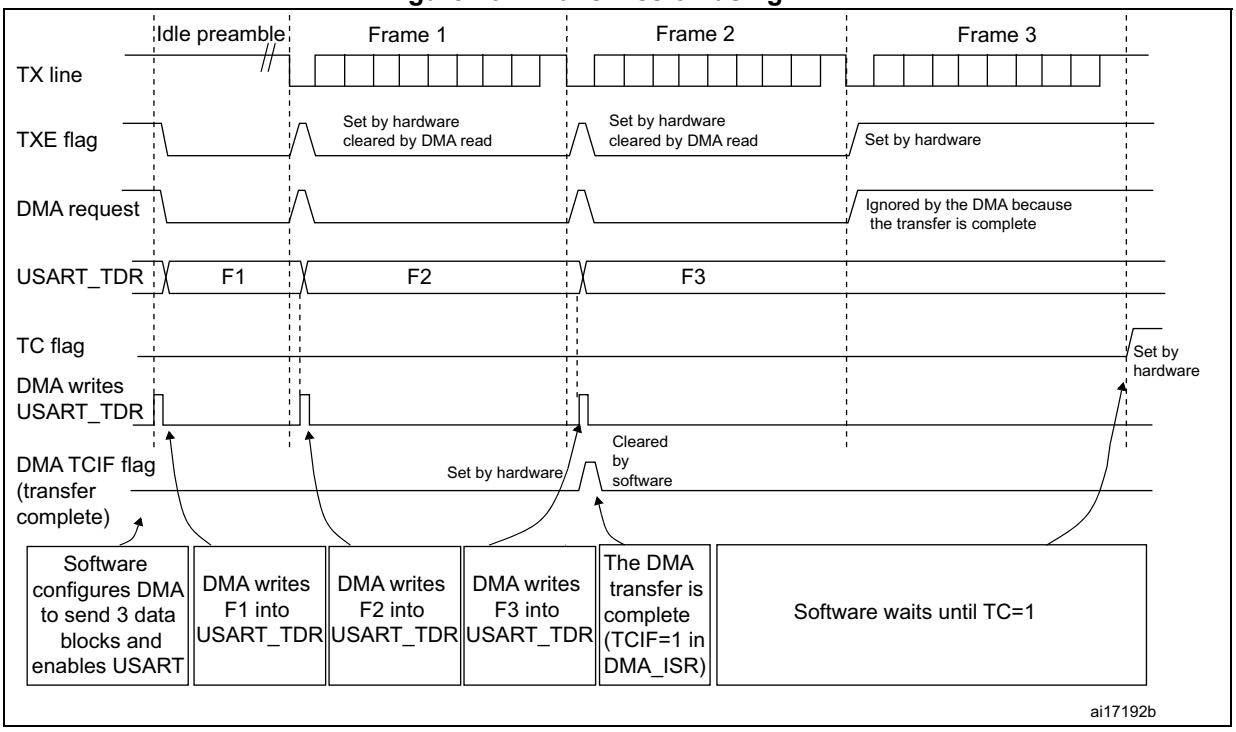
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode 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 462. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

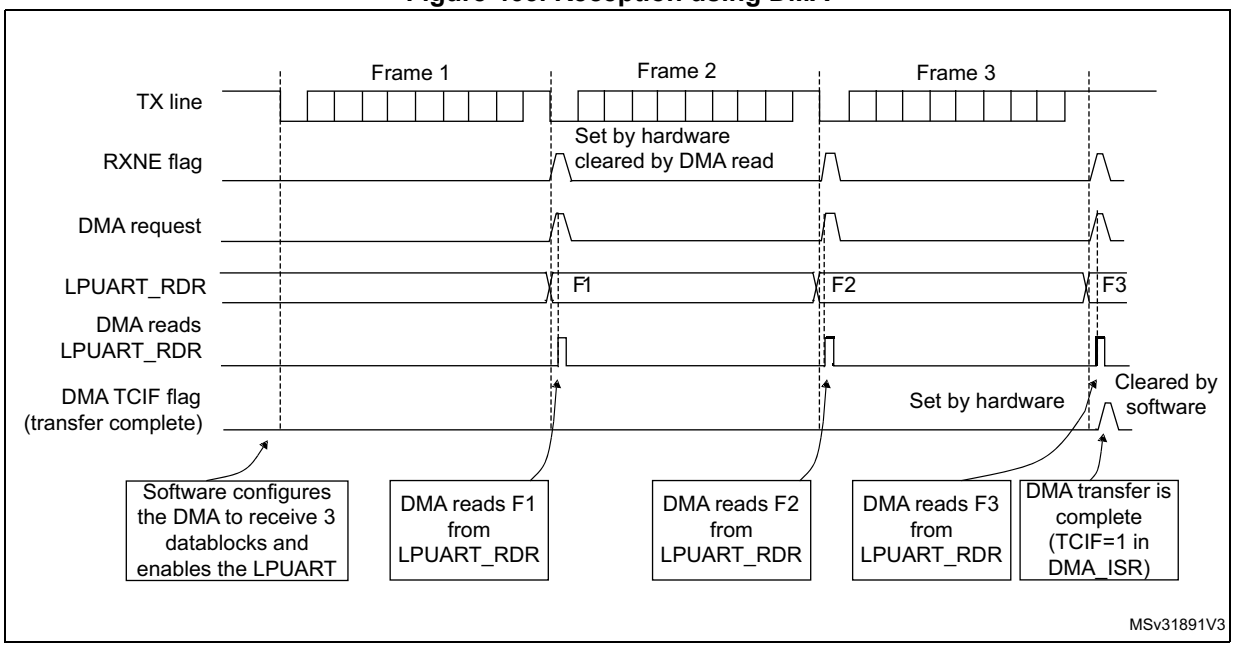
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller (DMA)* section) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 463. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

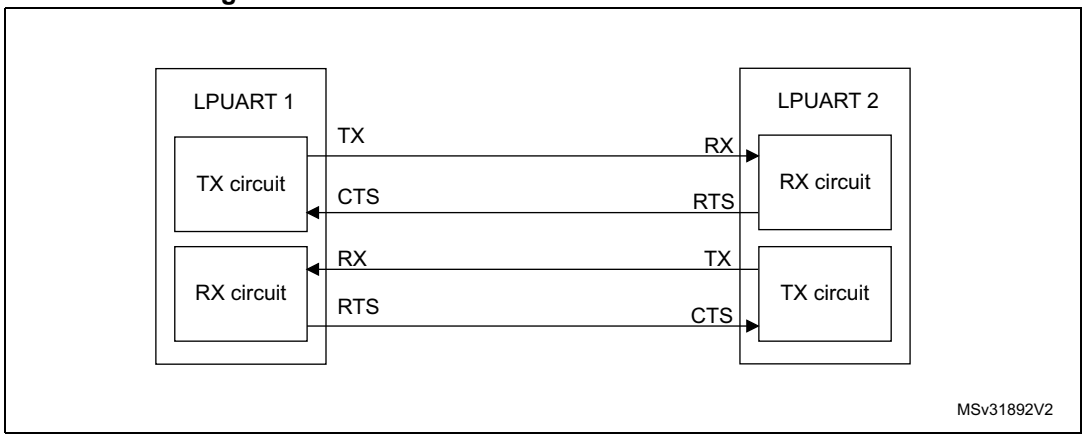
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt

enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

45.4.13 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 450* shows how to connect 2 devices in this mode:

Figure 464. Hardware flow control between 2 LPUARTs

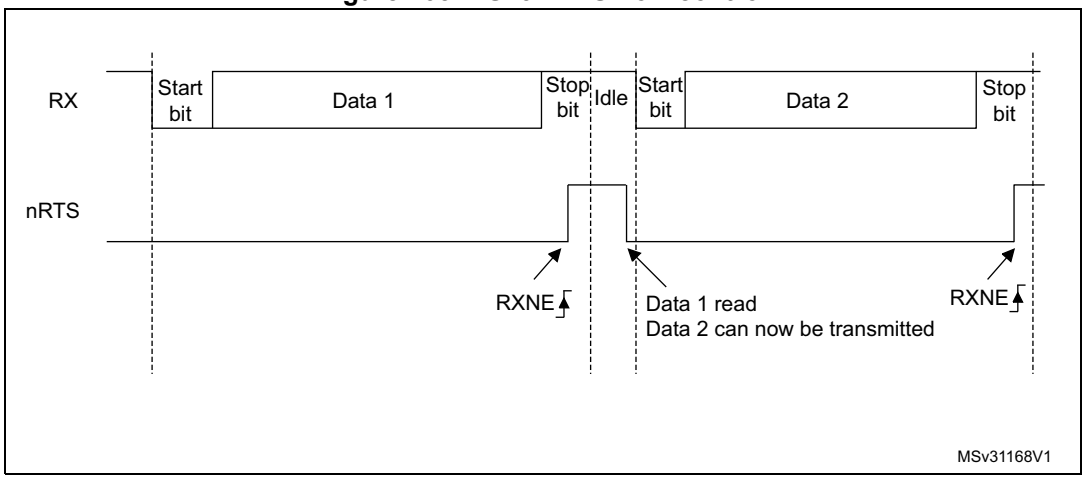


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

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 465* shows an example of communication with RTS flow control enabled.

Figure 465. RS232 RTS flow control



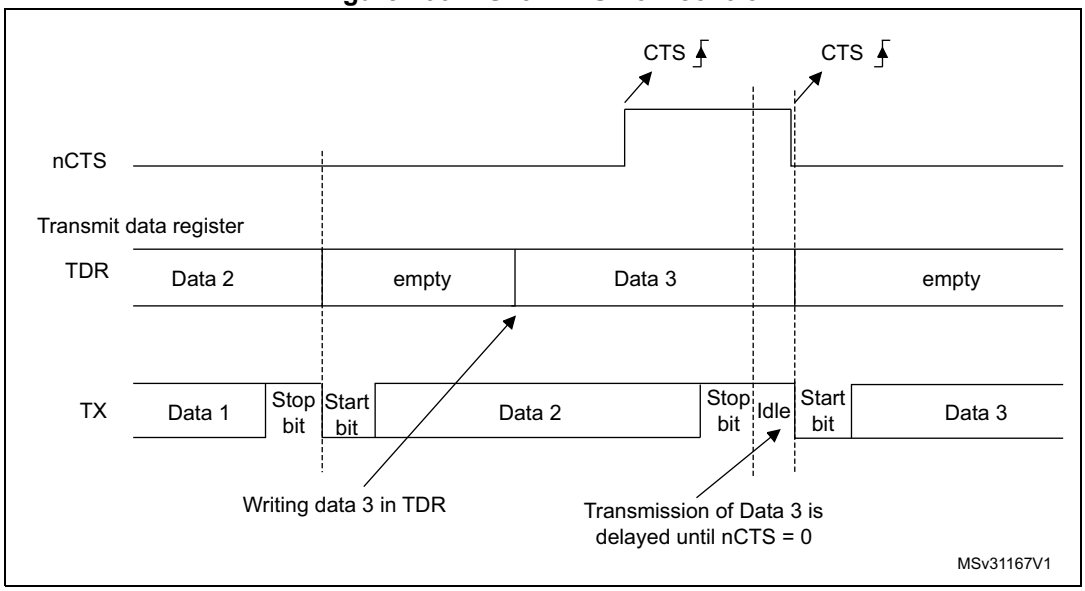
Note: When FIFO mode is enabled, nRTS is deasserted only when RXFIFO is full.

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 466* shows an example of communication with CTS flow control enabled.

Figure 466. 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]

45.4.14 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 467 shows the behavior when the wakeup event is verified.

Figure 467. Wakeup event verified (wakeup event = address match, FIFO disabled)

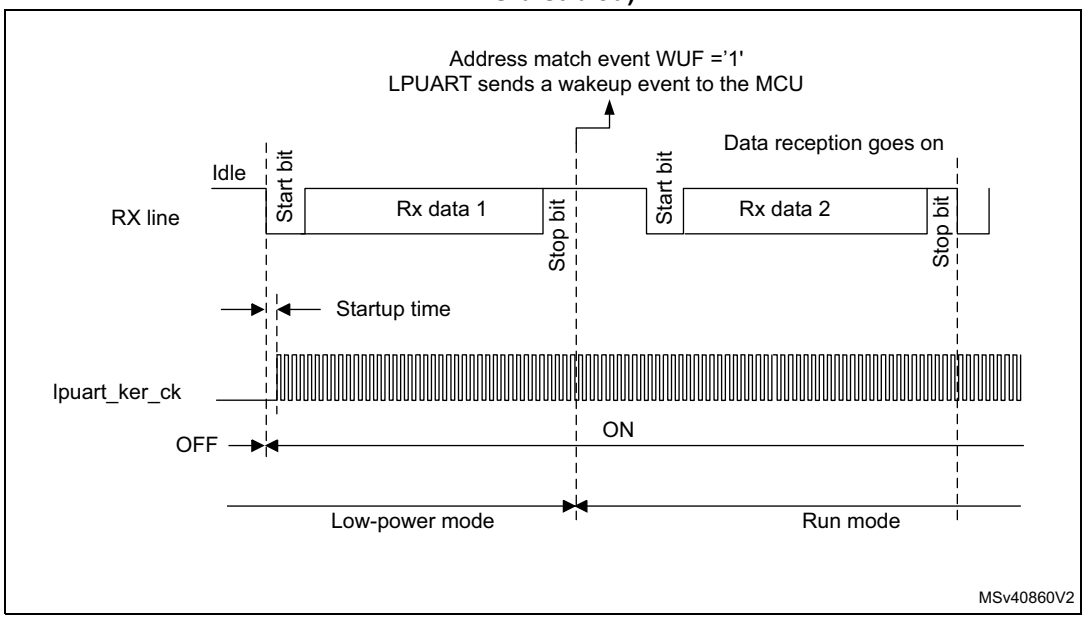
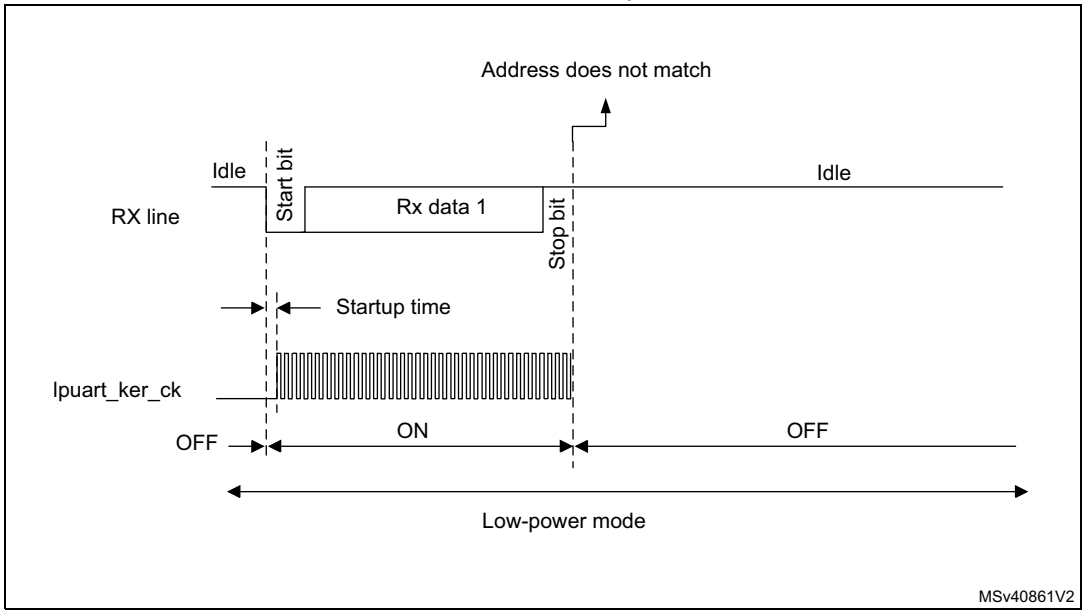


Figure 468 shows the behavior when the wakeup event is not verified.

Figure 468. 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 45.4.8: 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 348: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% - 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wakeup correctly from low-power mode is: 1/11.32 µs = 88.36 Kbaud.

45.5 LPUART in low-power modes

Table 350. 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 45.3: 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.

45.6 LPUART interrupts

Refer to *Table 351* for a detailed description of all LPUART interrupt requests.

Table 351. 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 45.3: 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.

45.7 LPUART registers

Refer to *Section 1.2 on page 76* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

45.7.1 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 44.5.20: 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 45.4.13: RS232 Hardware flow control and RS485 Driver Enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE/RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.

When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

45.7.2 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 44.5.20: 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 45.4.13: RS232 Hardware flow control and RS485 Driver Enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE/RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.

When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

45.7.3 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.

45.7.4 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 44.4: 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 44.4: 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.

45.7.5 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].

45.7.6 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.

45.7.7 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.

45.7.8 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

45.7.9 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 44.4: 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.

45.7.10 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 455*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

45.7.11 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 455*).

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.

45.7.12 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.

45.7.13 LPUART register map

The table below gives the LPUART register map and reset values.

Table 352. 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 352. 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 2.3: Memory organization* for the register boundary addresses.

46 Serial peripheral interface (SPI)

46.1 Introduction

The SPI interface can be used to communicate with external devices using the SPI protocol. SPI 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.

46.2 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

46.3 SPI implementation

The following table describes all the SPI instances and their features embedded in the devices.

Table 353. STM32L552xx and STM32L562xx SPI implementation

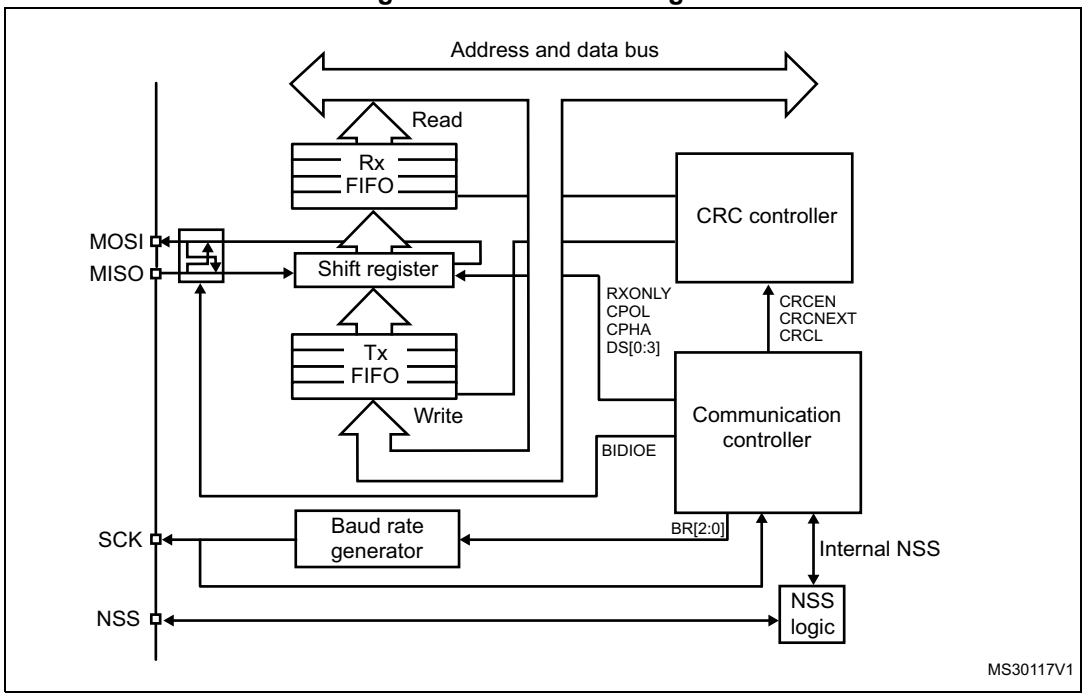
SPI features	SPI1	SPI2	SPI3
Enhanced NSSP & TI modes	Yes	Yes	Yes
Hardware CRC calculation	Yes	Yes	Yes
I2S support	No	No	No

46.4 SPI functional description

46.4.1 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 469*.

Figure 469. 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 46.4.5: 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.

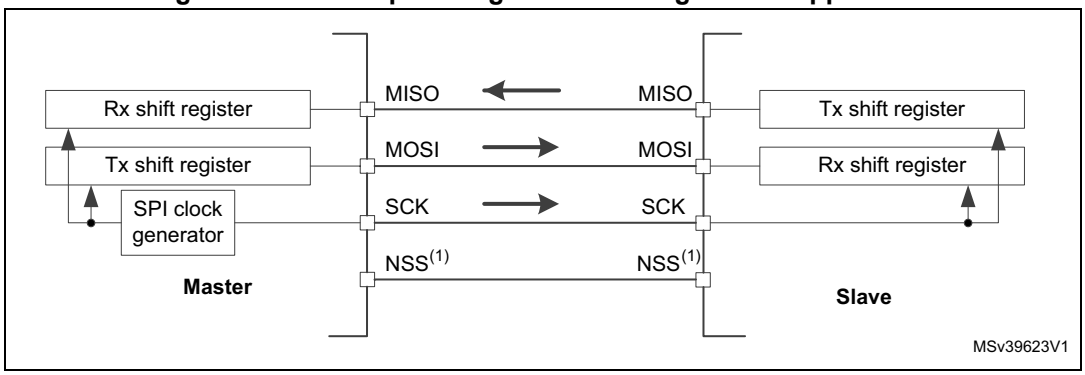
46.4.2 Communications between one master and one slave

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 470. Full-duplex single master/ single slave application

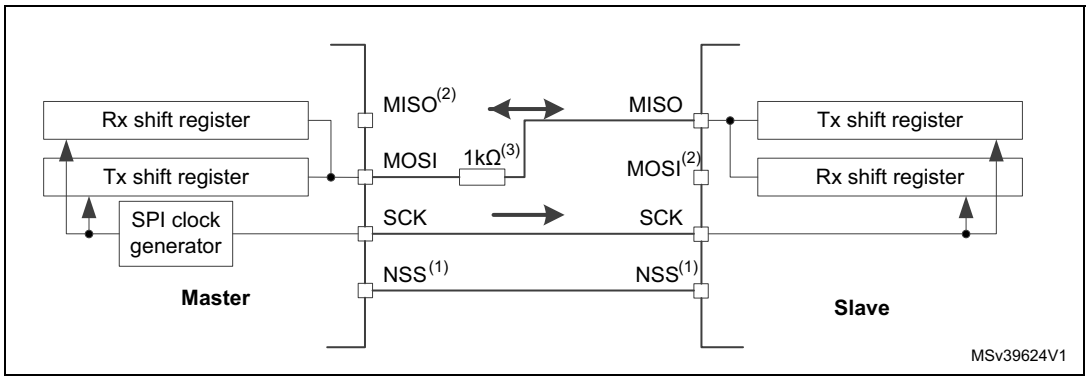


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 46.4.5: Slave select (NSS) pin management*.

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 471. Half-duplex single master/ single slave application



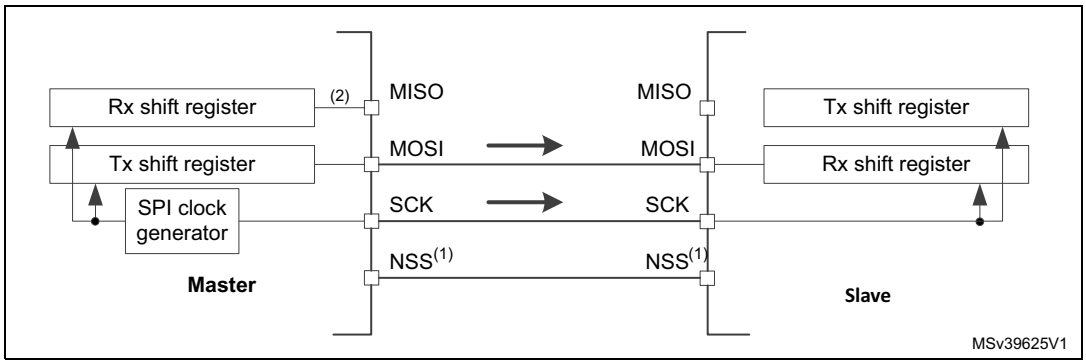
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 46.4.5: Slave select (NSS) pin management*.
2. In this 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 *46.4.5: 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 472. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)



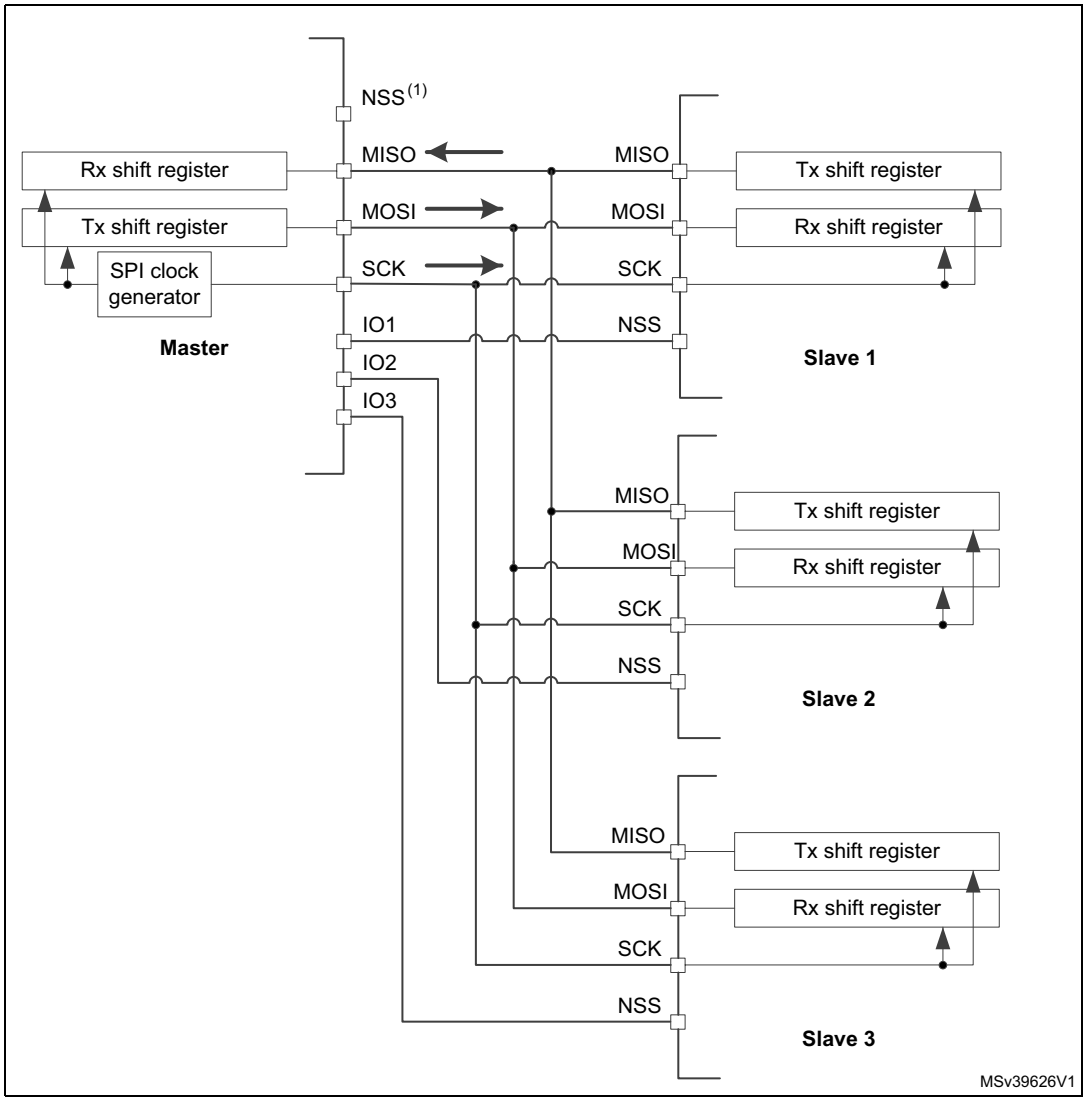
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 46.4.5: Slave select (NSS) pin management*.
2. An accidental 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).

46.4.3 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 473*). 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 473. 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)).

46.4.4 Multi-master communication

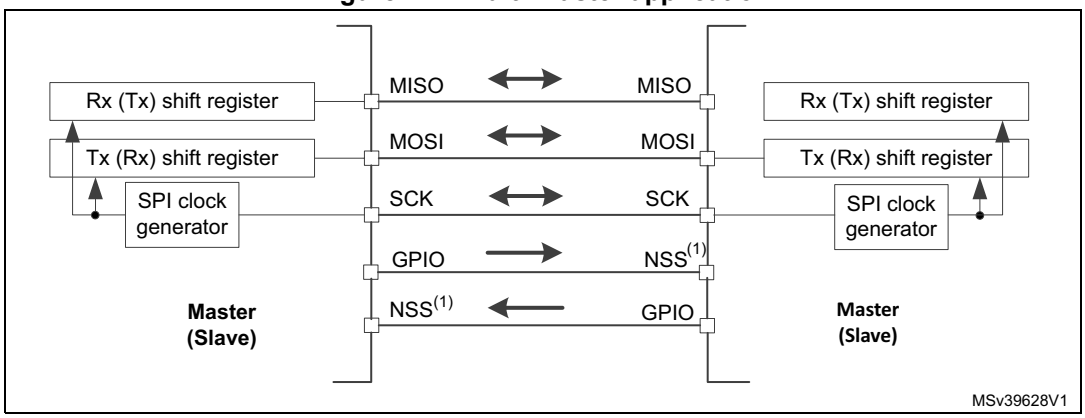
Unless SPI bus is not designed for a multi-master capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

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 474. 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.

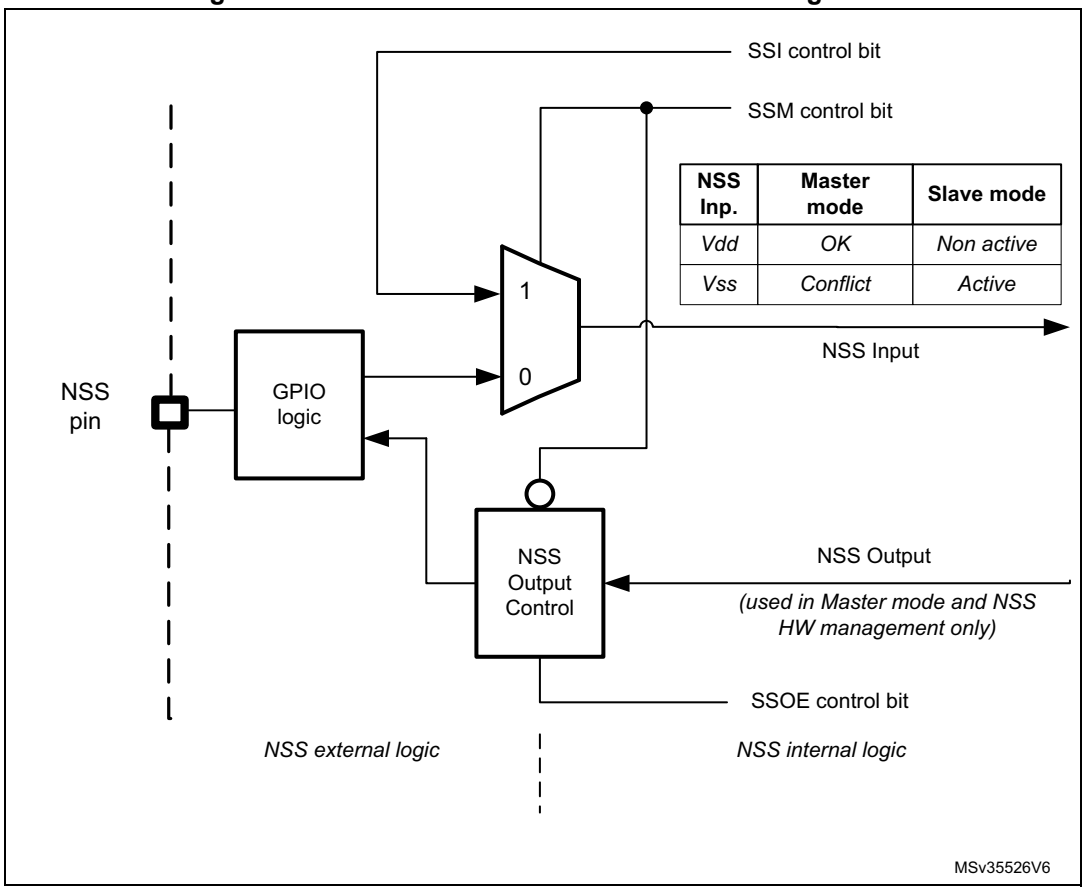
46.4.5 Slave select (NSS) pin management

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 475. Hardware/software slave select management



46.4.6 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.

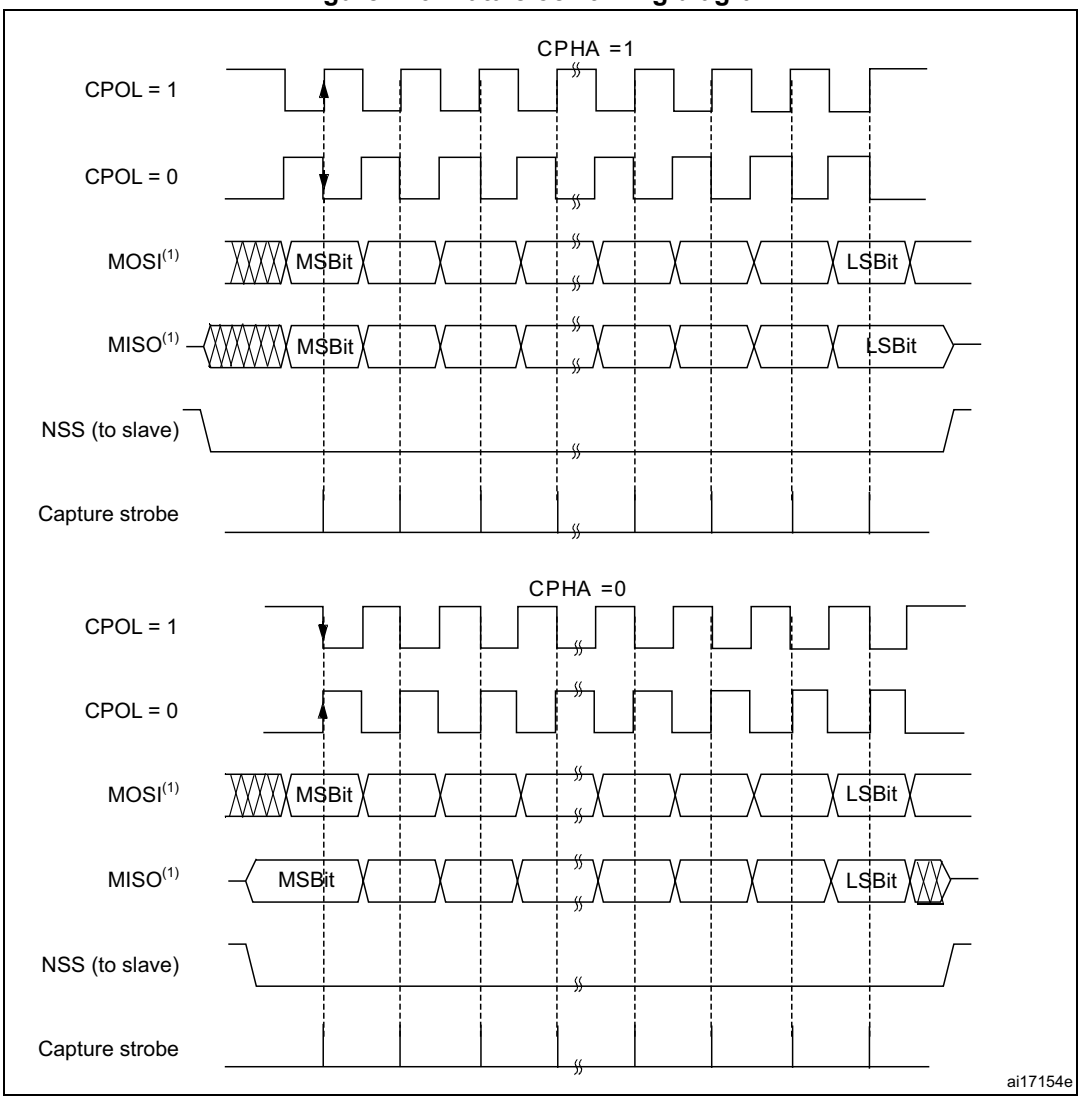
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 476, 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 476. Data clock timing diagram

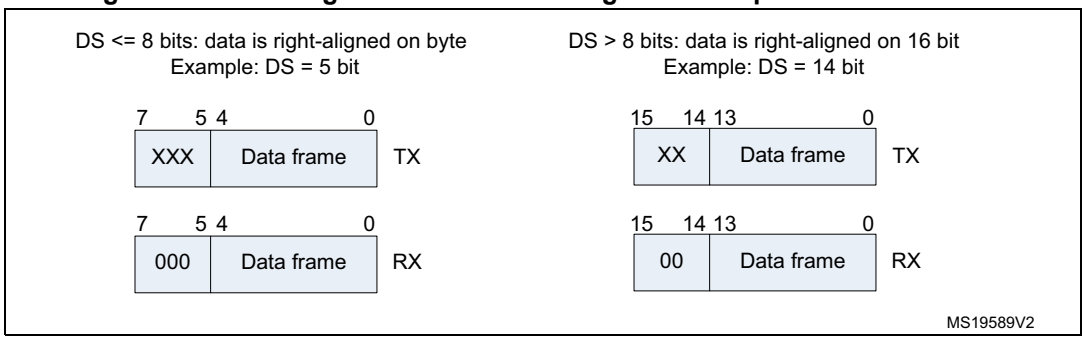


1. The order of data bits depends on LSBFIRST bit setting.

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 477*). During communication, only bits within the data frame are clocked and transferred.

Figure 477. 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.

46.4.7 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

46.4.8 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.

46.4.9 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 46.4.14: 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 46.4.13: 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 479* through *Figure 482*.

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 46.4.10: 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 46.4.5: 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.

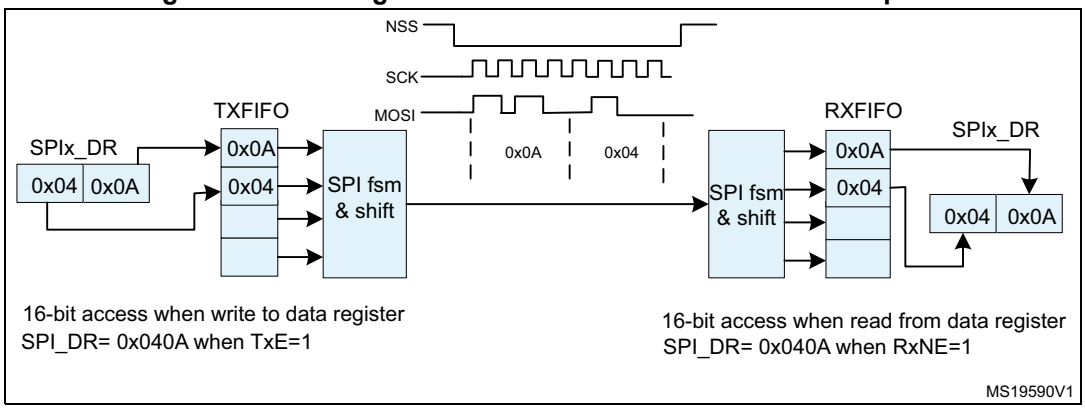
Data packing

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 478* 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 478. Packing data in FIFO for transmission and reception



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 479* through *Figure 482*.

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 1711*.)

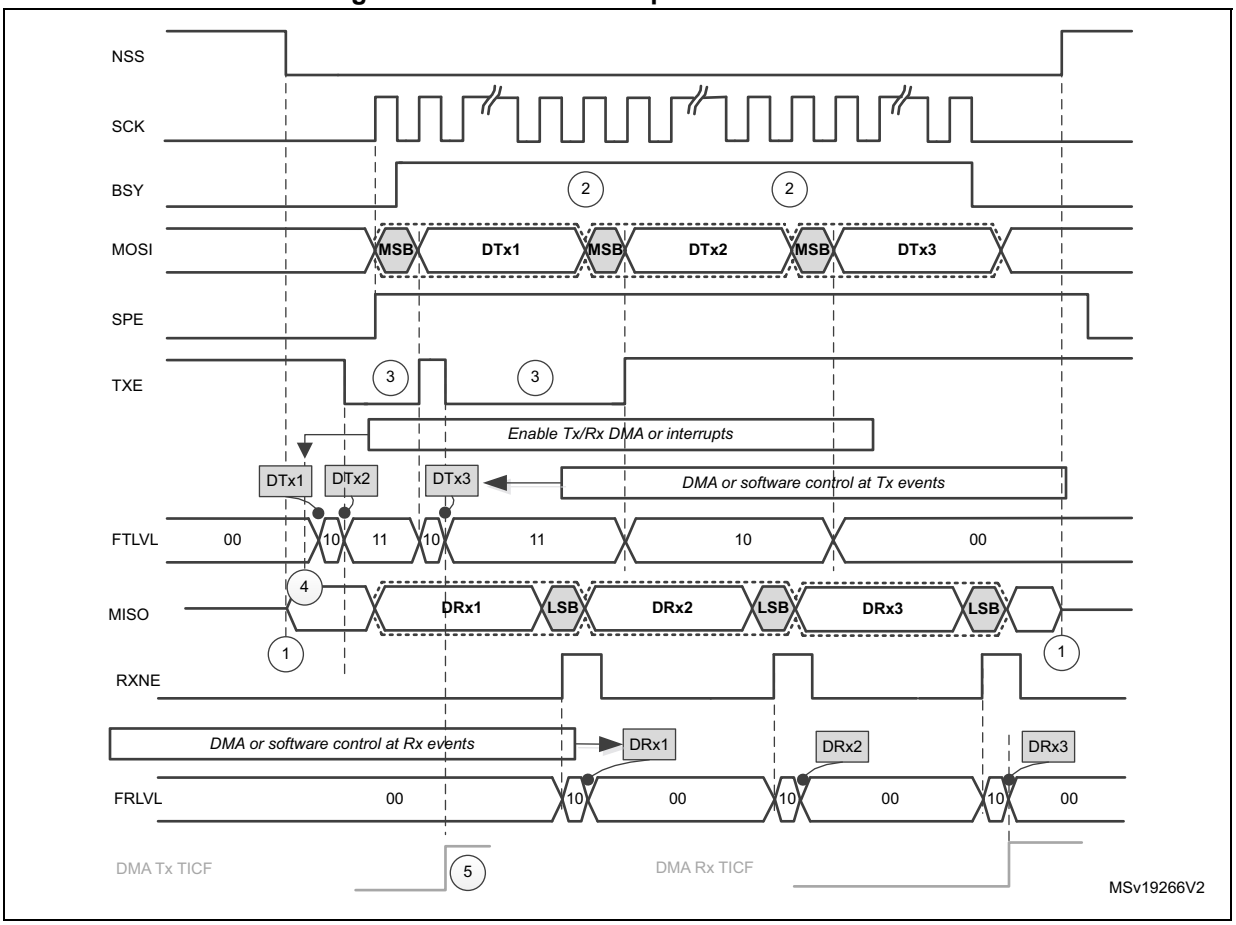
Communication diagrams

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 479 on page 1715* through *Figure 482 on page 1718*:

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 479. Master full-duplex communication



Assumptions for master full-duplex communication example:

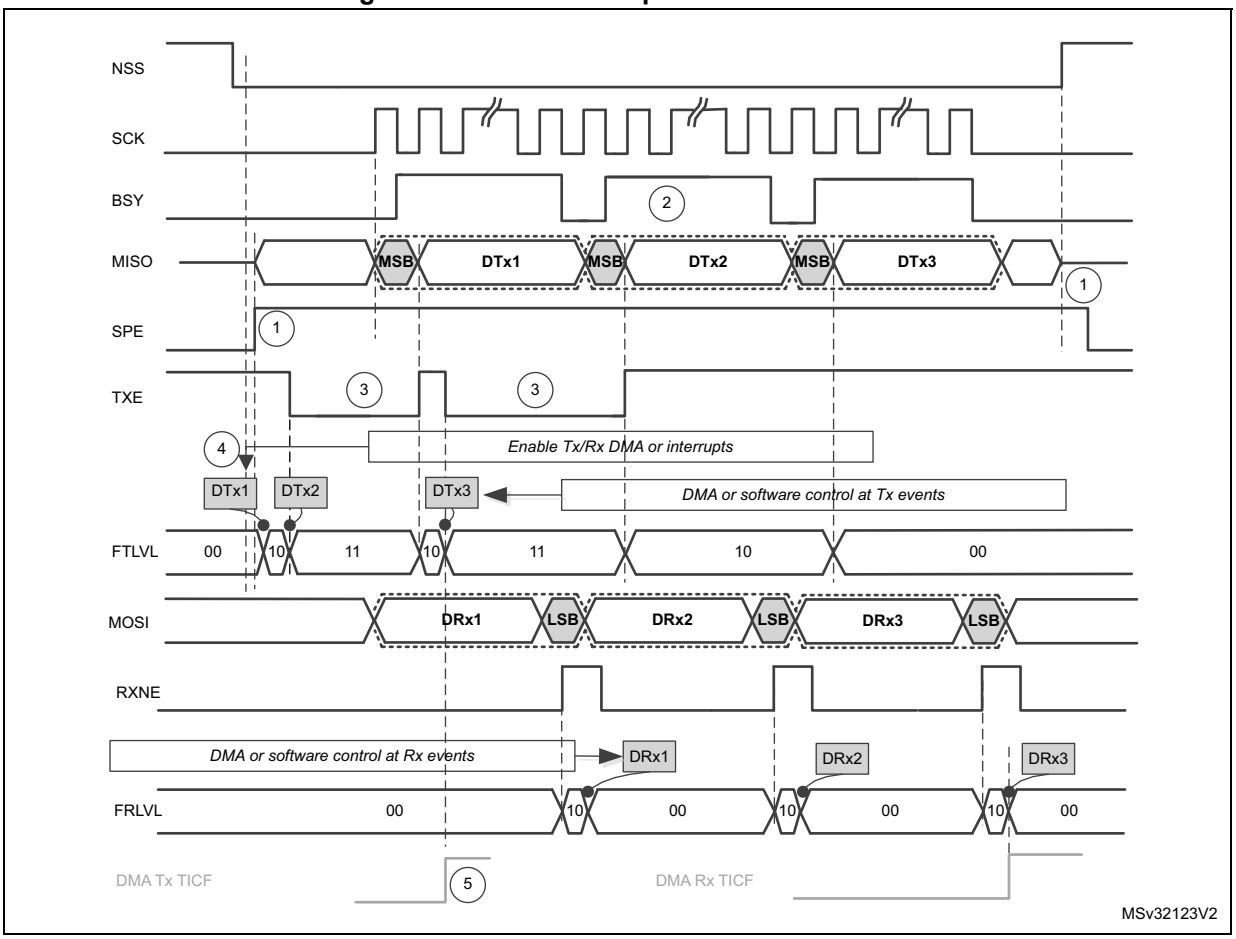
- Data size > 8 bit

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 1714* for details about common assumptions and notes.

Figure 480. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

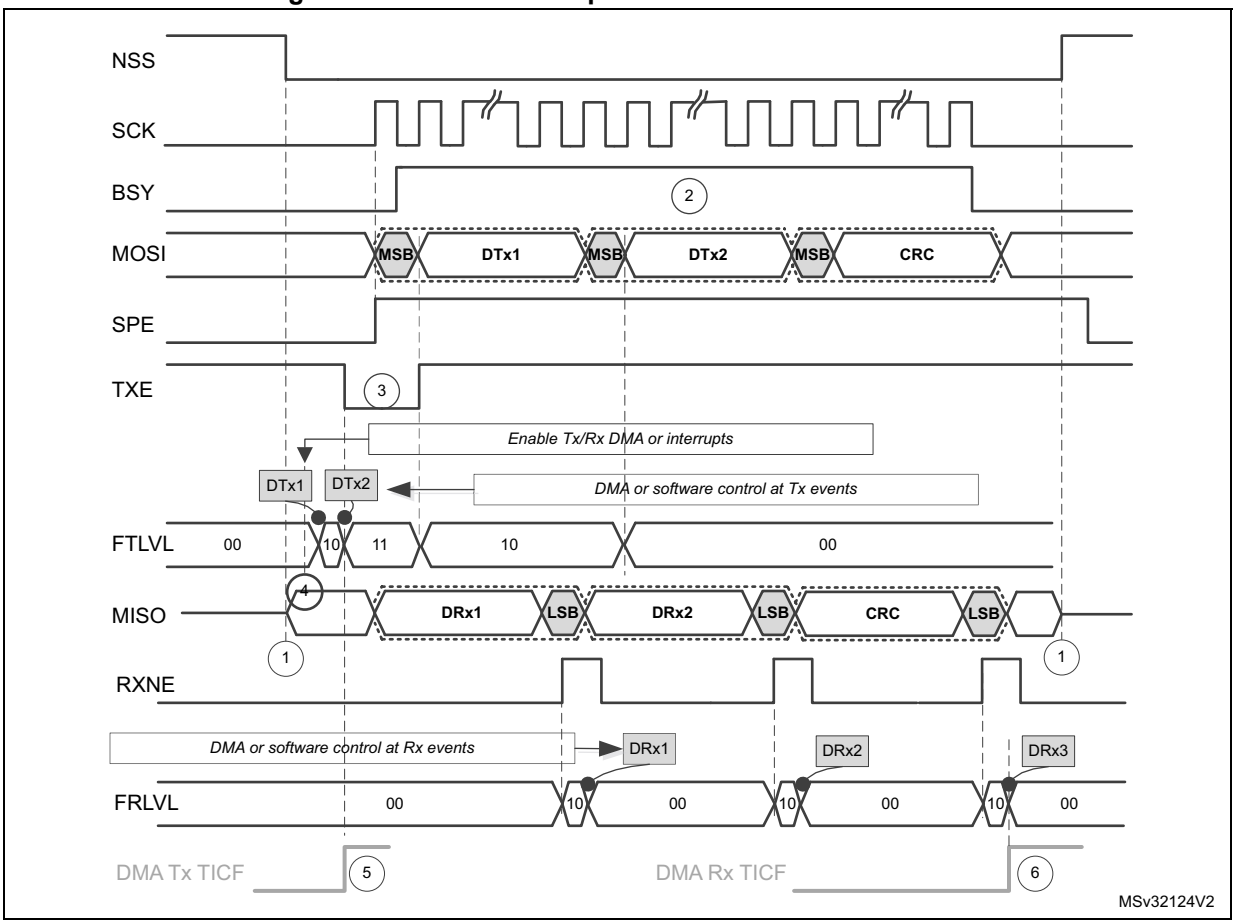
- Data size > 8 bit

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 1714* for details about common assumptions and notes.

Figure 481. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

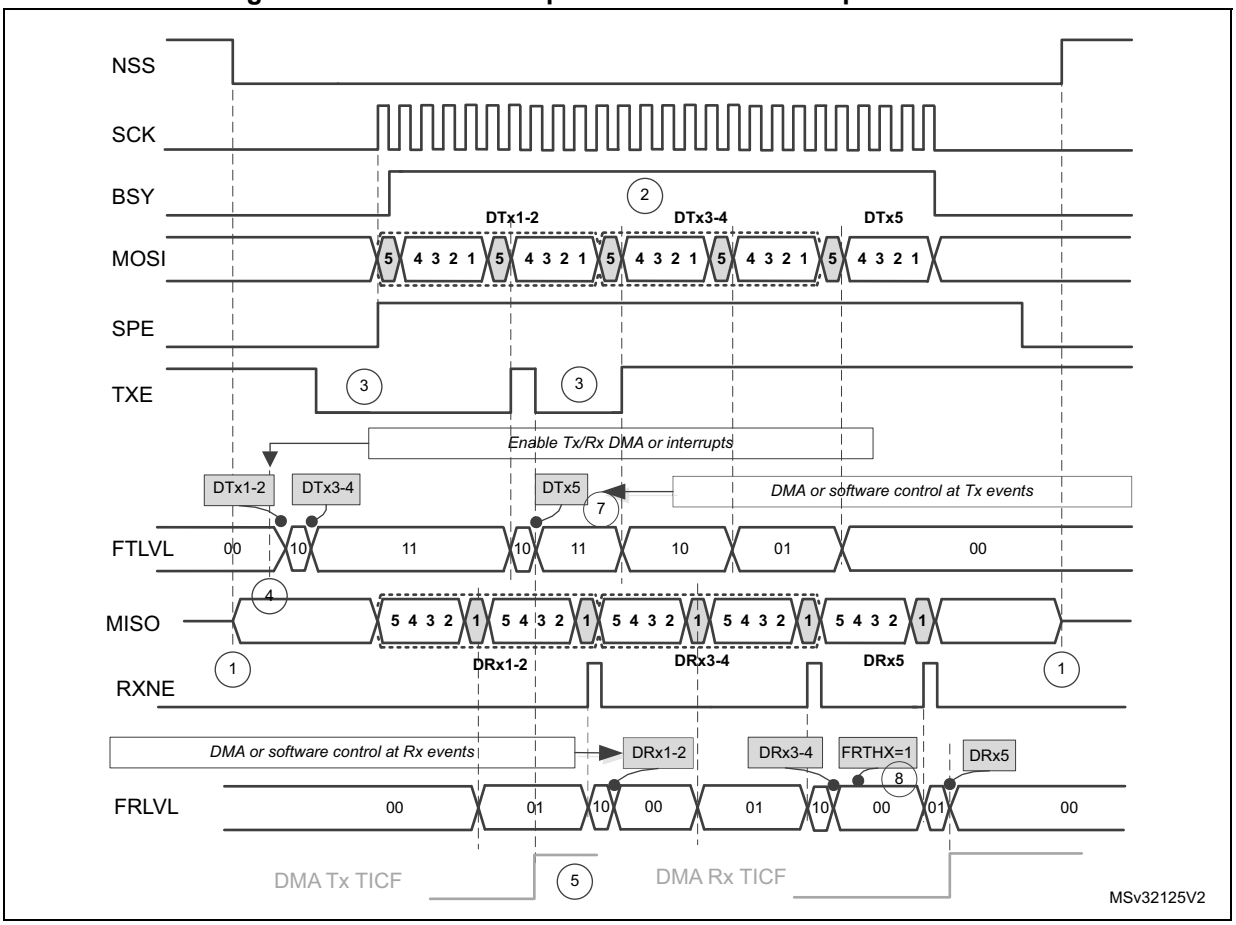
- Data size = 16 bit
- CRC enabled

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 1714* for details about common assumptions and notes.

Figure 482. 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 1714* for details about common assumptions and notes.

46.4.10 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.

46.4.11 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 46.4.14: 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.

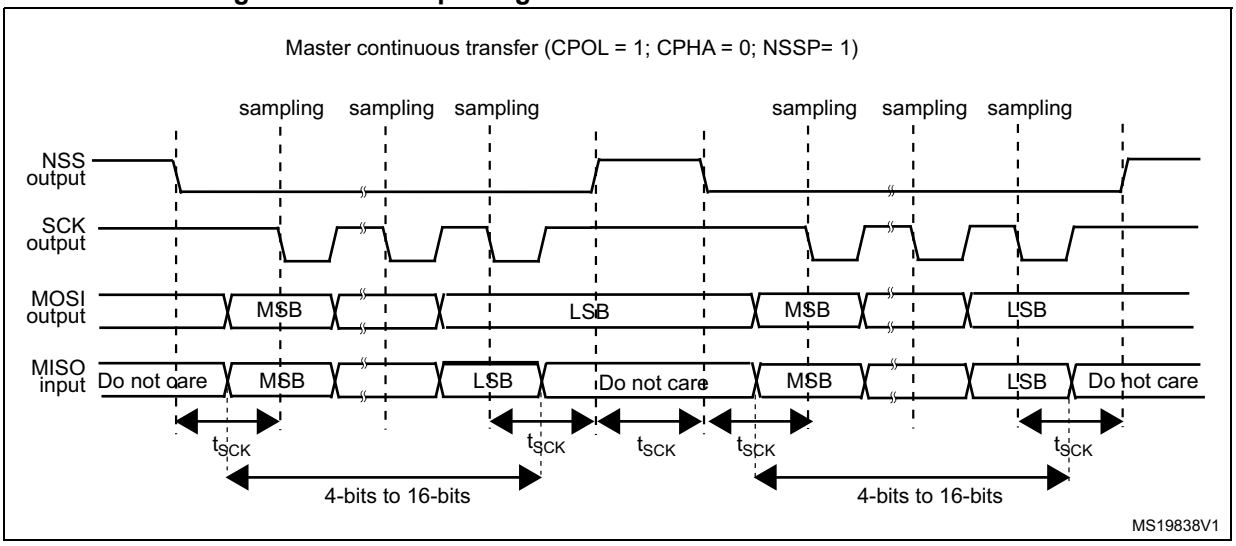
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

46.4.12 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 483 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 483. 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.

46.4.13 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 484*). 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:

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

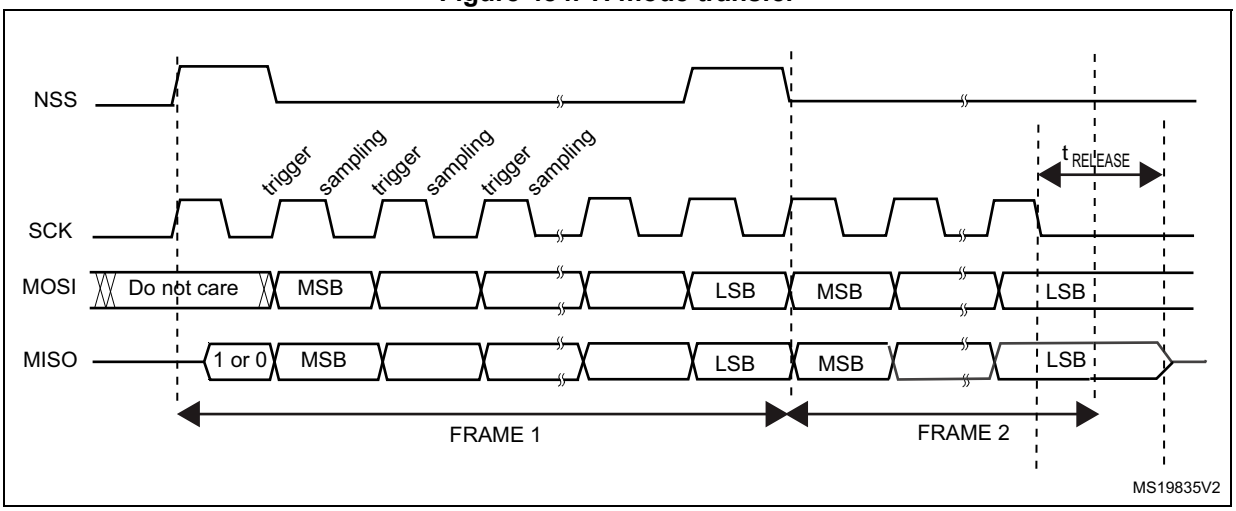
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

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 484: TI mode transfer shows the SPI communication waveforms when TI mode is selected.

Figure 484. TI mode transfer



46.4.14 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.

46.5 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 354. 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	

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.

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.

46.6 SPI 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.

46.6.1 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

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.

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.

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.

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.

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)

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 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 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 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 1710.

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.

Bit 2 **MSTR:** Master selection

0: Slave configuration

1: Master configuration

Note: This bit should not be changed when communication is ongoing.

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 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 SPI TI mode except the case when CRC is applied at TI mode.

46.6.2 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 1710 if the CRCEN bit is set.

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 1710 if the CRCEN bit is set.

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)

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)

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).

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).

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 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 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

46.6.3 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	Res.	Res.	TXE	RXNE
			r	r	r	r	r	r	r	r	rc_w0			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)

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 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. Refer to *Section 46.4.11: SPI 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 not busy
1: SPI 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 46.4.10: SPI status flags and Procedure for disabling the SPI on page 1710.

Bit 6 **OVR:** Overrun flag
0: No overrun occurred
1: Overrun occurred
This flag is set by hardware and reset by a 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 1720* for the software sequence.

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.

Bits 3:2 Reserved, must be kept at reset value.

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

46.6.4 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 46.4.9: 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.

46.6.5 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.

46.6.6 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.

A read to this register when the BSY Flag is set could return an incorrect value.

46.6.7 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.

A read to this register when the BSY flag is set could return an incorrect value.

46.6.8 SPI register map

Table 355 shows the SPI register map and reset values.

Table 355. SPI 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	Res.	Res.	TXE	RXNE
	Reset value																				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

Refer to *Section 2.3 on page 87* for the register boundary addresses.

47 Serial audio interface (SAI)

47.1 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).

The SAI can be connected with other SAIs to work synchronously.

47.2 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.
- Possible synchronization between multiple SAIs.
- 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.

47.3 SAI implementation

Table 356. STM32L5 Series SAI features (1)

SAI features	SAI1	SAI2
I2S, LSB or MSB-justified, PCM/DSP, TDM, AC'97	X	X
FIFO size	8 words	8 words
SPDIF	X	X
PDM	X(2)	-

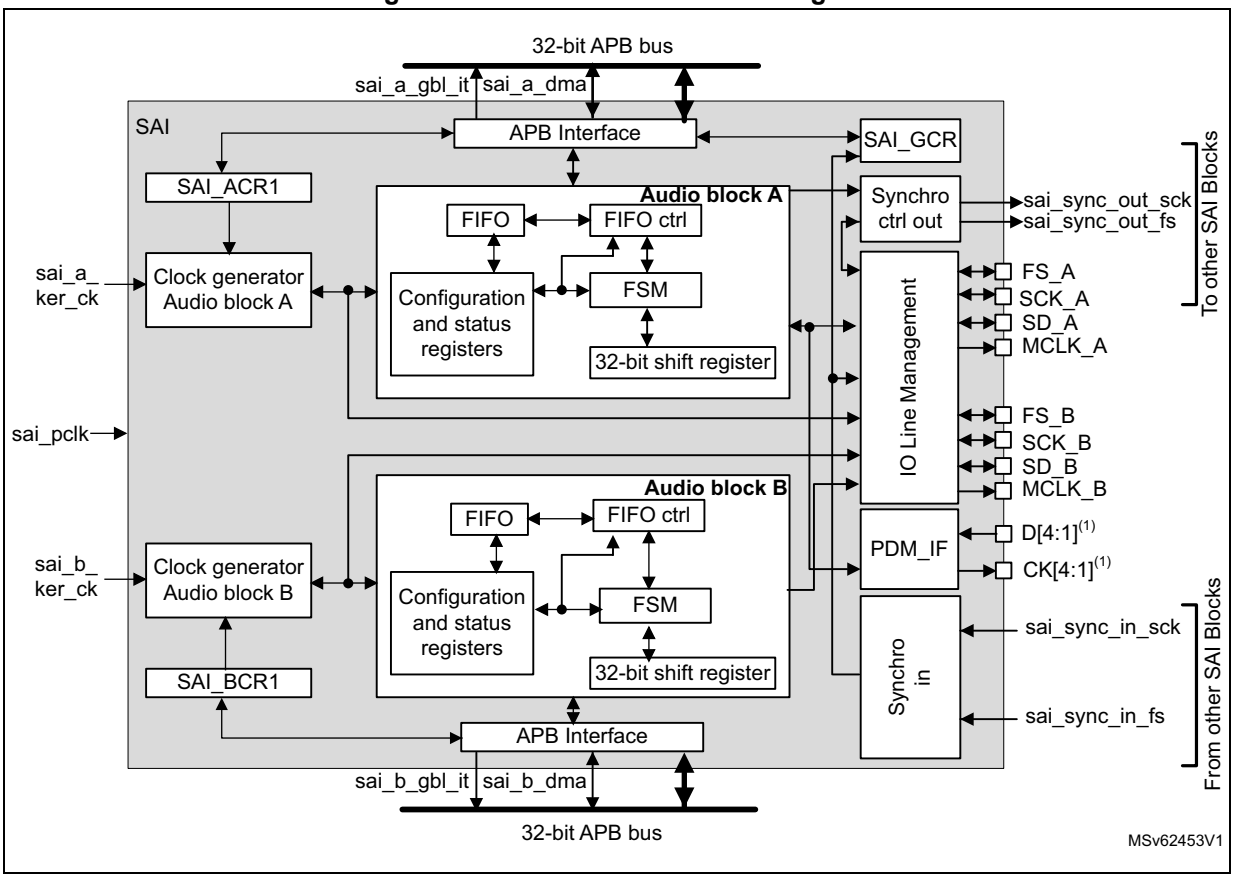
1. 'X' = supported, '-' = not supported.
2. Only signals D[3:1], and CK[2:1] are available.

47.4 SAI functional description

47.4.1 SAI block diagram

Figure 485 shows the SAI block diagram while *Table 357* and *Table 358* list SAI internal and external signals.

Figure 485. SAI functional block diagram



1. These signals might not be available for all SAI instances. Refer to *Section 47.3: SAI implementation* for details.

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 ise 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.

47.4.2 SAI pins and internal signals

Table 357. 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_sync_out_sck, sai_sync_out_fs	Output	Internal clock and frame synchronization output signals exchanged with other SAI blocks.
sai_sync_in_sck, sai_sync_in_fs	Input	Internal clock and frame synchronization input signals exchanged with other SAI blocks.
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock.
sai_pclk	Input	APB clock.

Table 358. 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 47.3: SAI implementation* for details.

47.4.3 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 another SAI interface or 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.

47.4.4 SAI synchronization mode

There are two levels of synchronization, either at audio subblock level or at SAI level.

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.

External synchronization

The audio subblocks can also be configured to operate synchronously with another SAI. This can be done as follow:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio subblock is supposed to provide the FS and SCK signals to other SAI. This is done by programming SYNCOUT[1:0] bits.
2. The SAI which receives the synchronization signals, has to select which SAI provides the synchronization by setting the proper value on SYNCIN[1:0] bits. For each of the two SAI audio subblocks, the user must then specify if it operates synchronously with the other SAI via the SYNCEN bit.

Note: SYNCIN[1:0] and SYNCOUT[1:0] bits are located into the SAI_GCR register, and SYNCEN bits into SAI_xCR1 register.

If both audio subblocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the SYNCEN[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the SYNCEN[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through SYNCEN[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCIN must be set to 1. Positions noted as ‘Reserved’ must not be used.

Table 359. External synchronization selection

Block instance	SYNCIN= 1	SYNCIN= 0
SAI1	SAI2 sync.	Reserved
SAI2	Reserved	SAI1 sync.

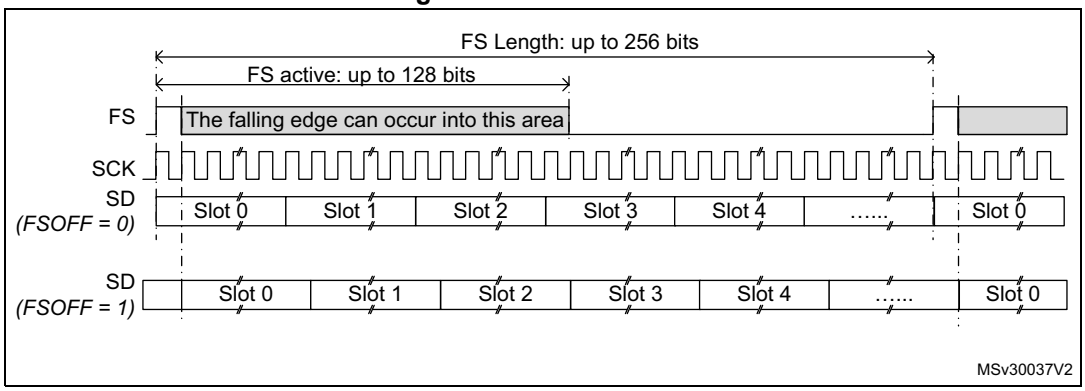
47.4.5 Audio data size

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.

47.4.6 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 486* illustrates this flexibility.

Figure 486. 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 47.4.8: 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 47.4.14: 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 47.4.14: 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 47.4.14: 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.

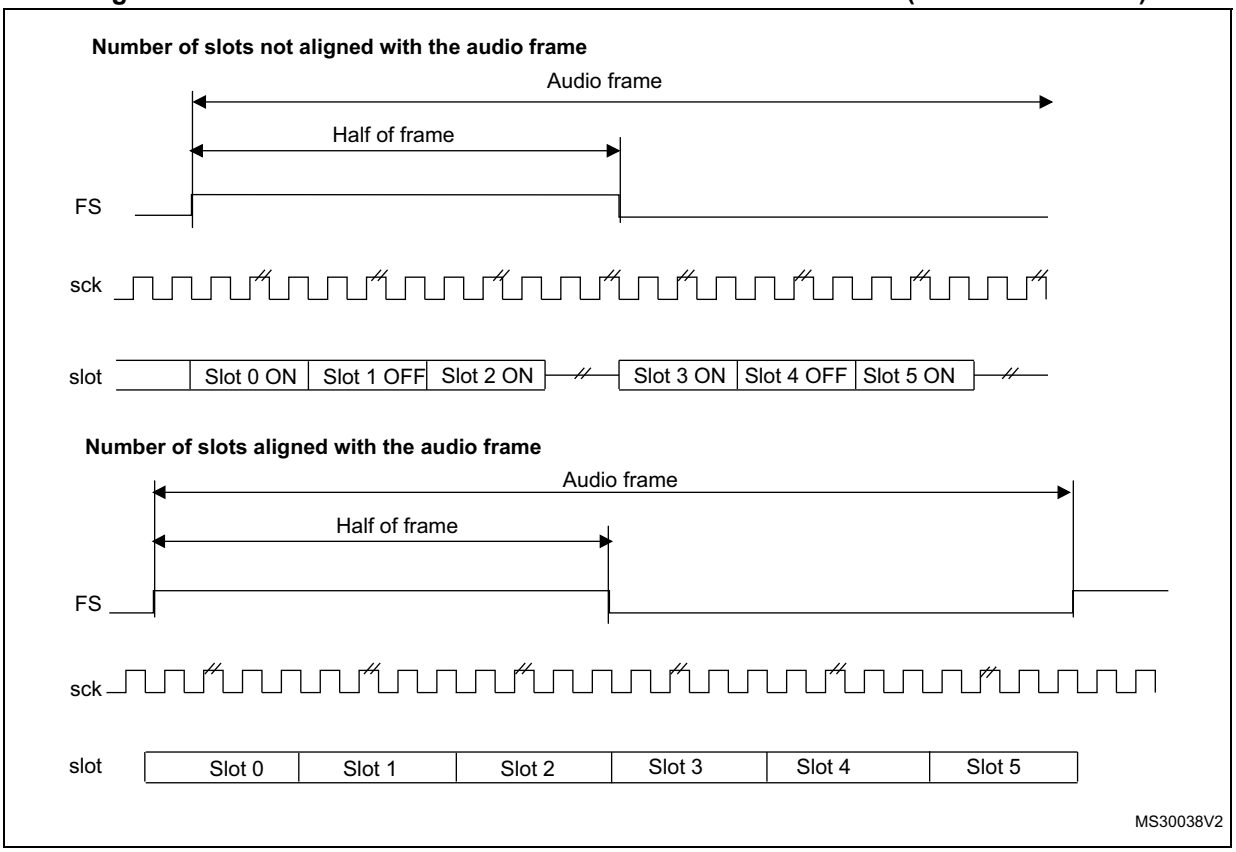
FS signal role

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 487. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

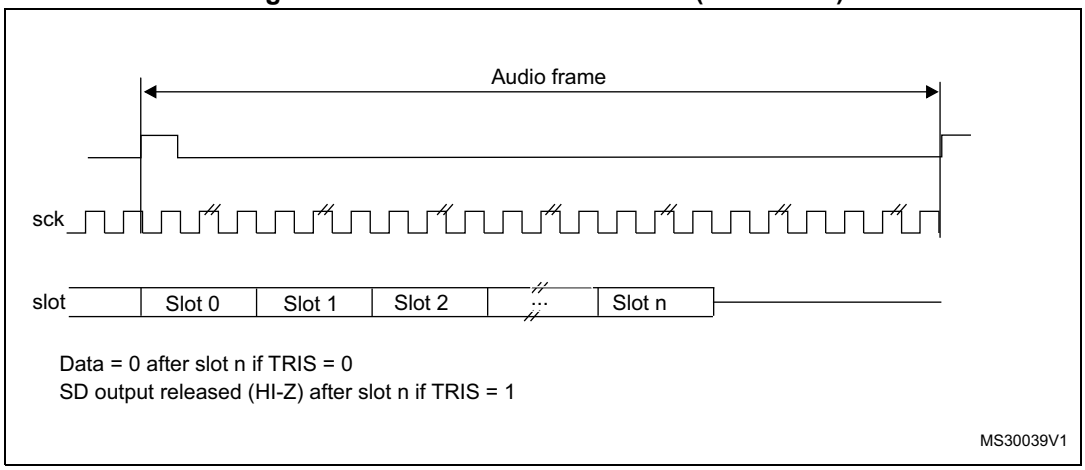


1. The frame length should be even.

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 488. FS role is start of frame (FSDEF = 0)



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.

47.4.7 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.

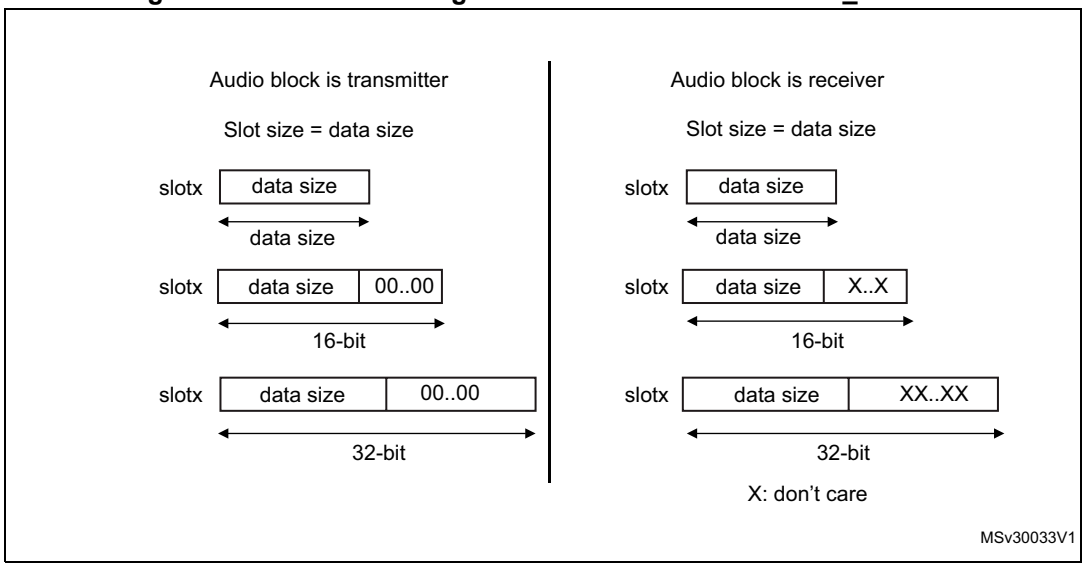
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

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.

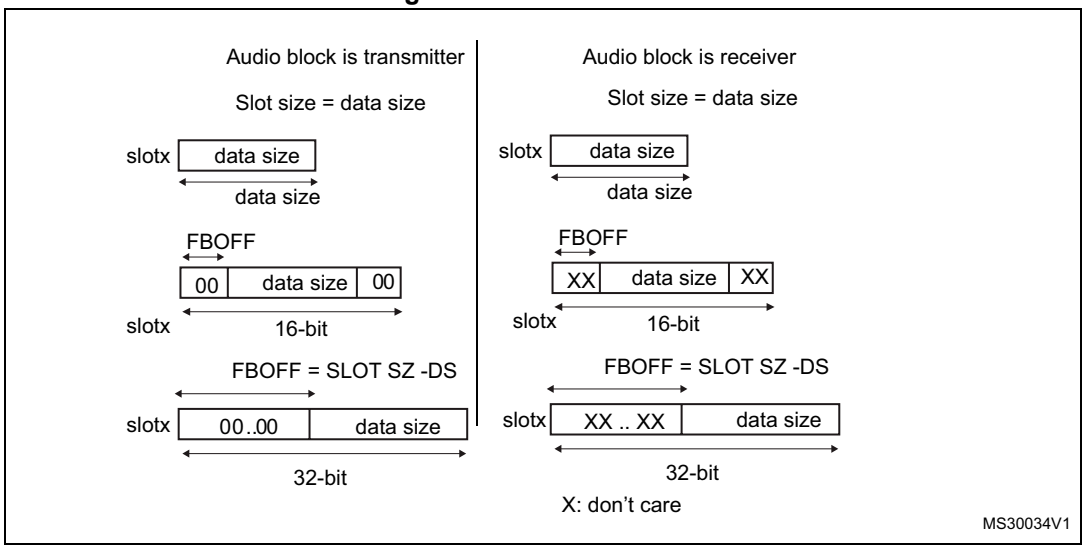
The slot size is also configurable as shown in *Figure 489*. The size of the slots is selected by setting SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 489. 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 490. 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 47.4.11: AC'97 link controller*.

47.4.8 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 360 shows MCLK_x activation conditions.

Table 360. 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)

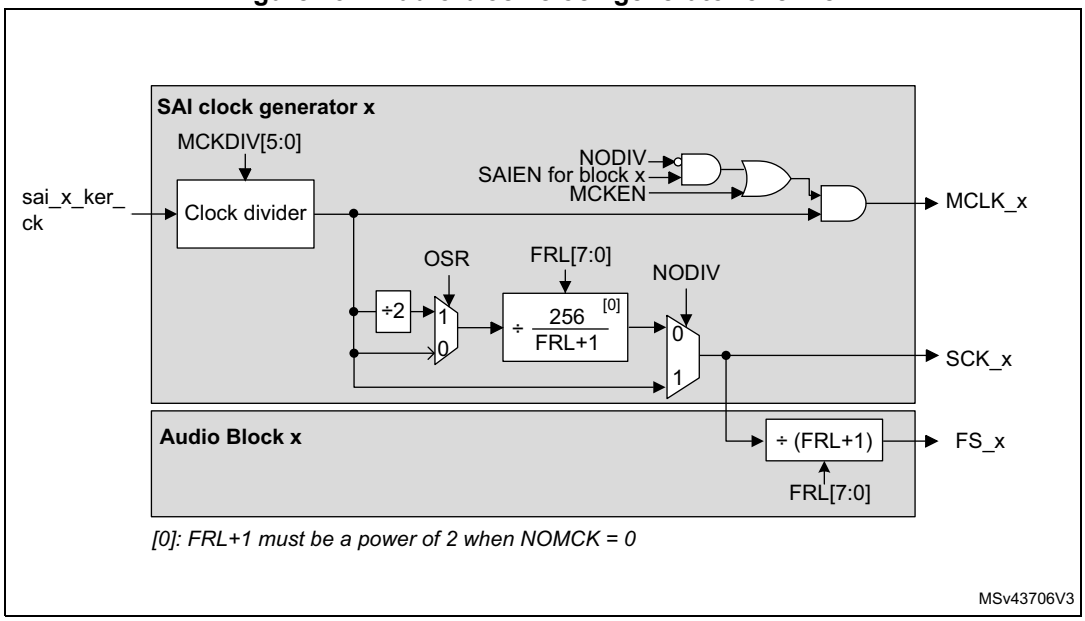
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

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 491 illustrates the architecture of the audio block clock generator.

Figure 491. 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:

- $F_{MCLK\_x} = 256 \times F_{FS\_x}$ if OSR = 0
- $F_{MCLK\_x} = 512 \times F_{FS\_x}$ 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 361 gives programming examples for 48, 96 and 192 kHz.

Table 361. 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.

47.4.9 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.

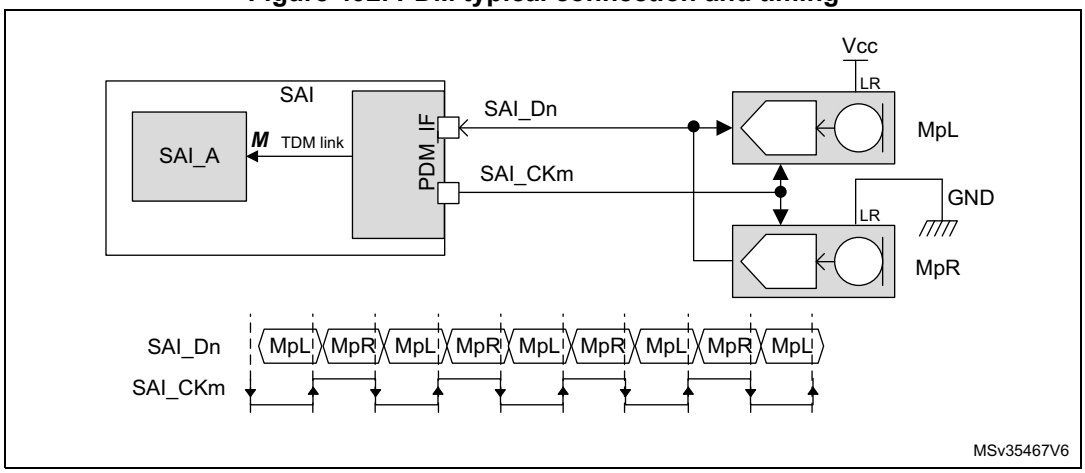
Data received are right aligned in the SAI_xDR.

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.

47.4.10 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 492* 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 492. 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]).

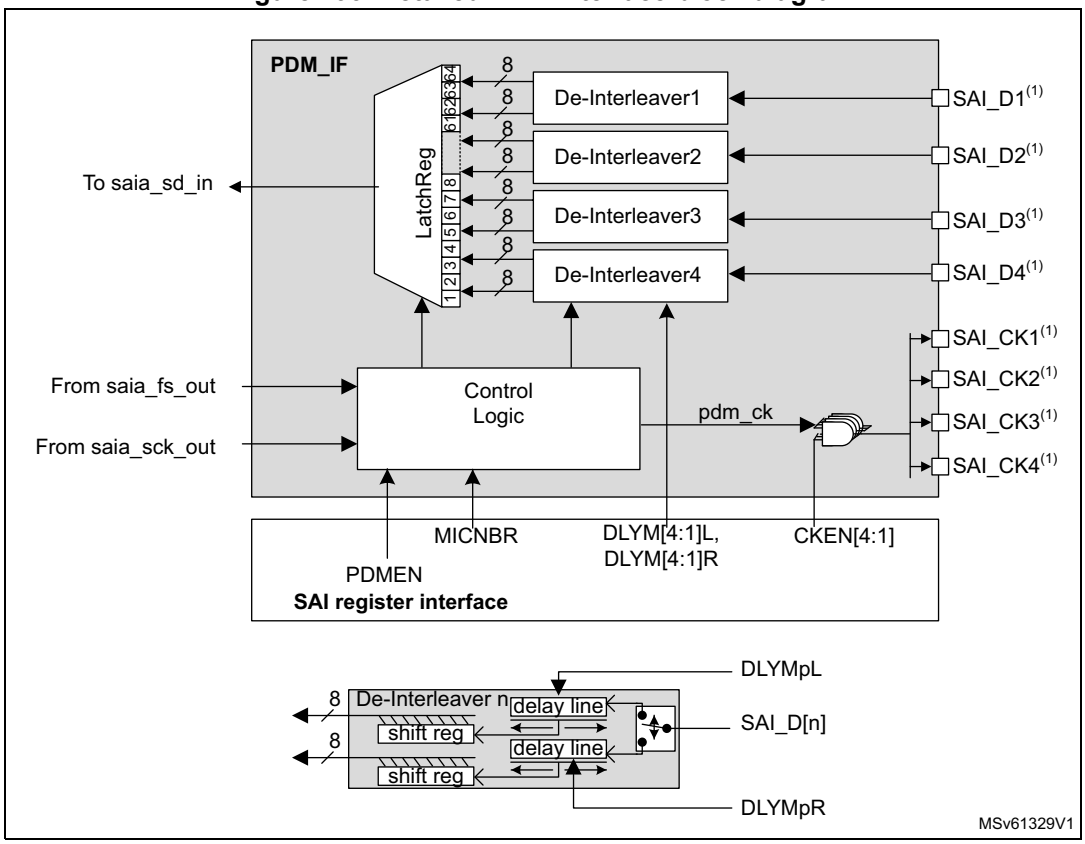
The data processing sequence into the PDM is the following:

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 493 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 493. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.
2. These signals might not be available in all SAI instances. Please refer to *Section 47.3: SAI implementation* for details.

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:

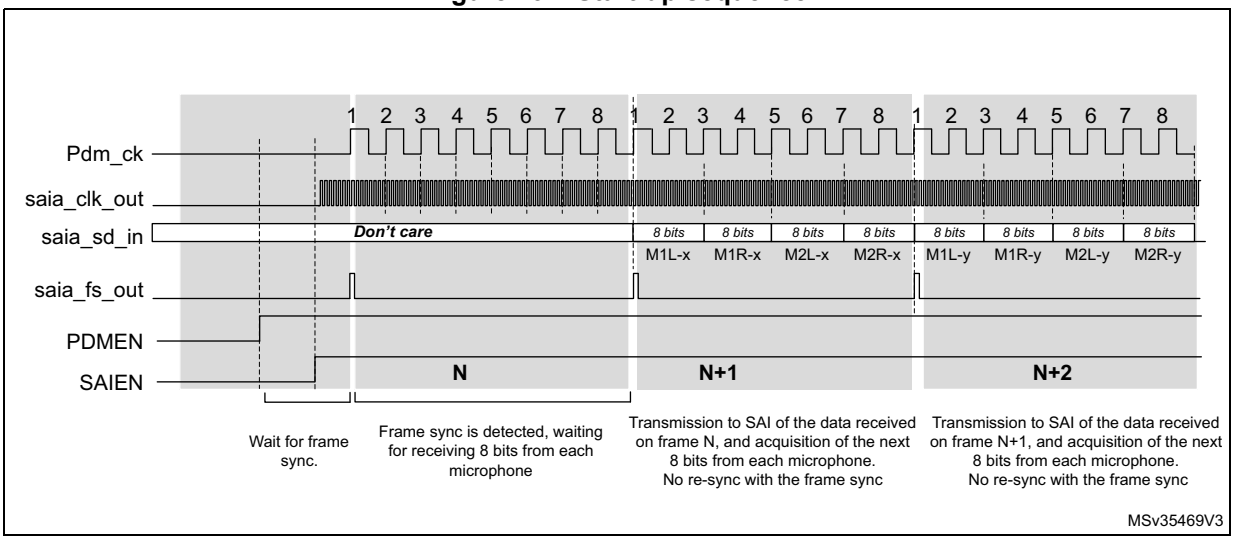
1. Configure SAI_A in TDM master mode (see *Table 362*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via PDMEN 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 494 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 494. Start-up sequence



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.

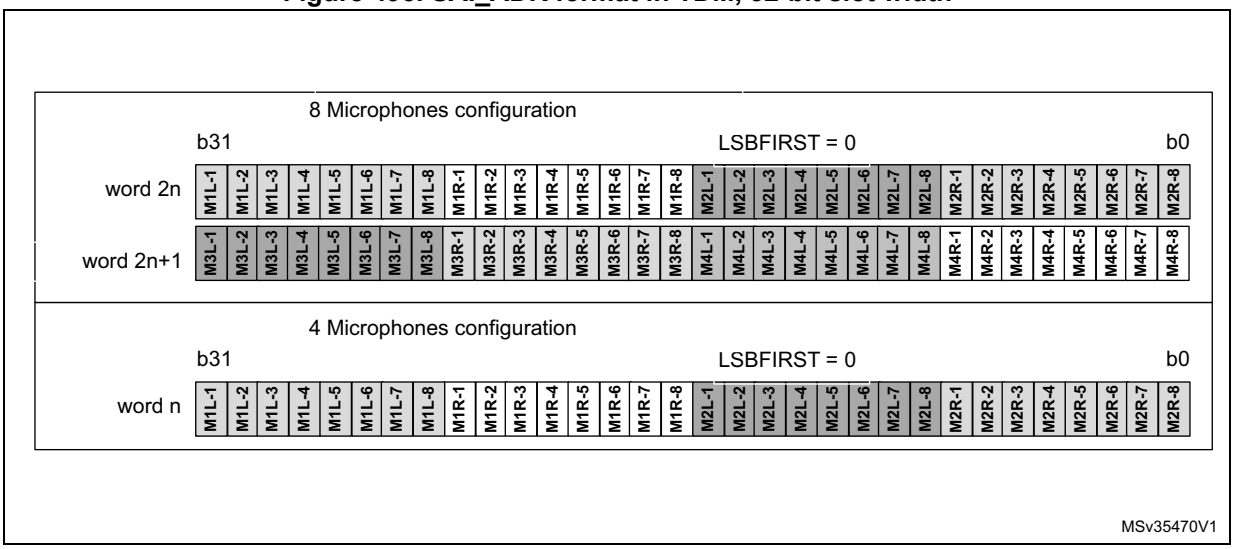
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 495*.

  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 495. SAI_ADR format in TDM, 32-bit slot width

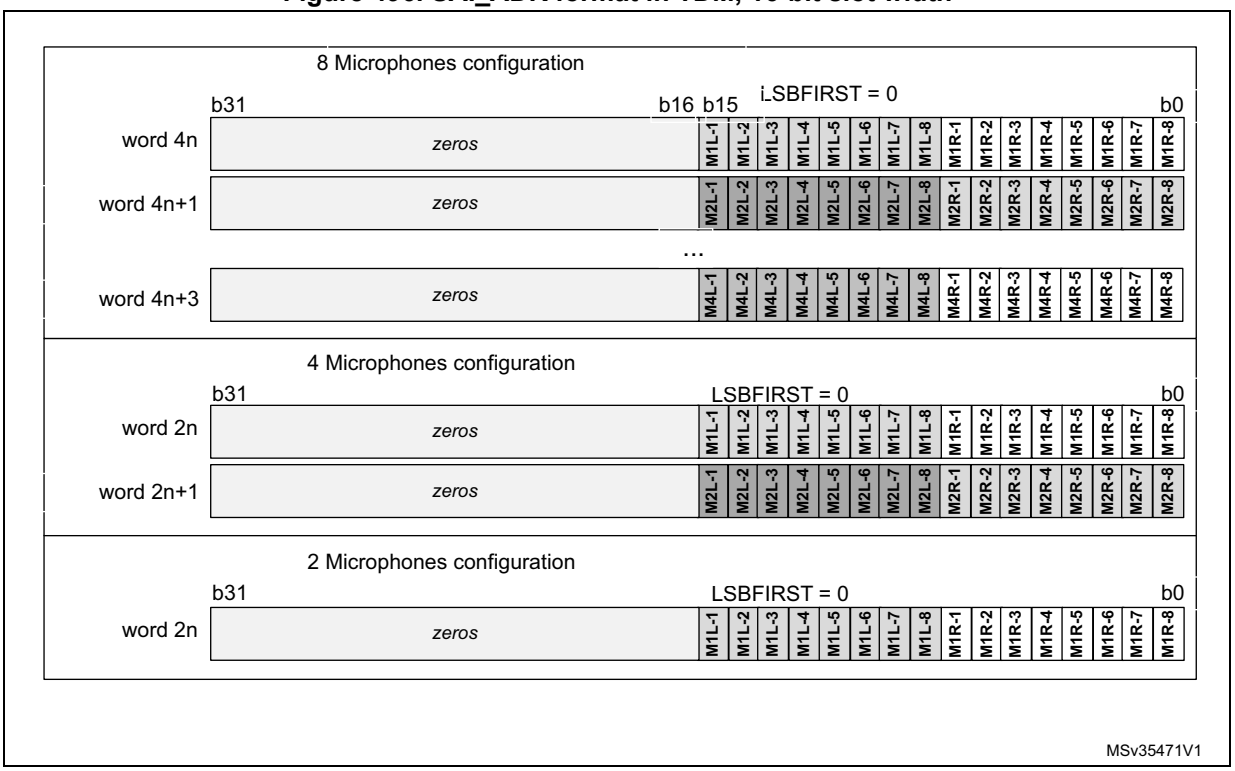


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 496*.

  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 496. SAI_ADR format in TDM, 16-bit slot width

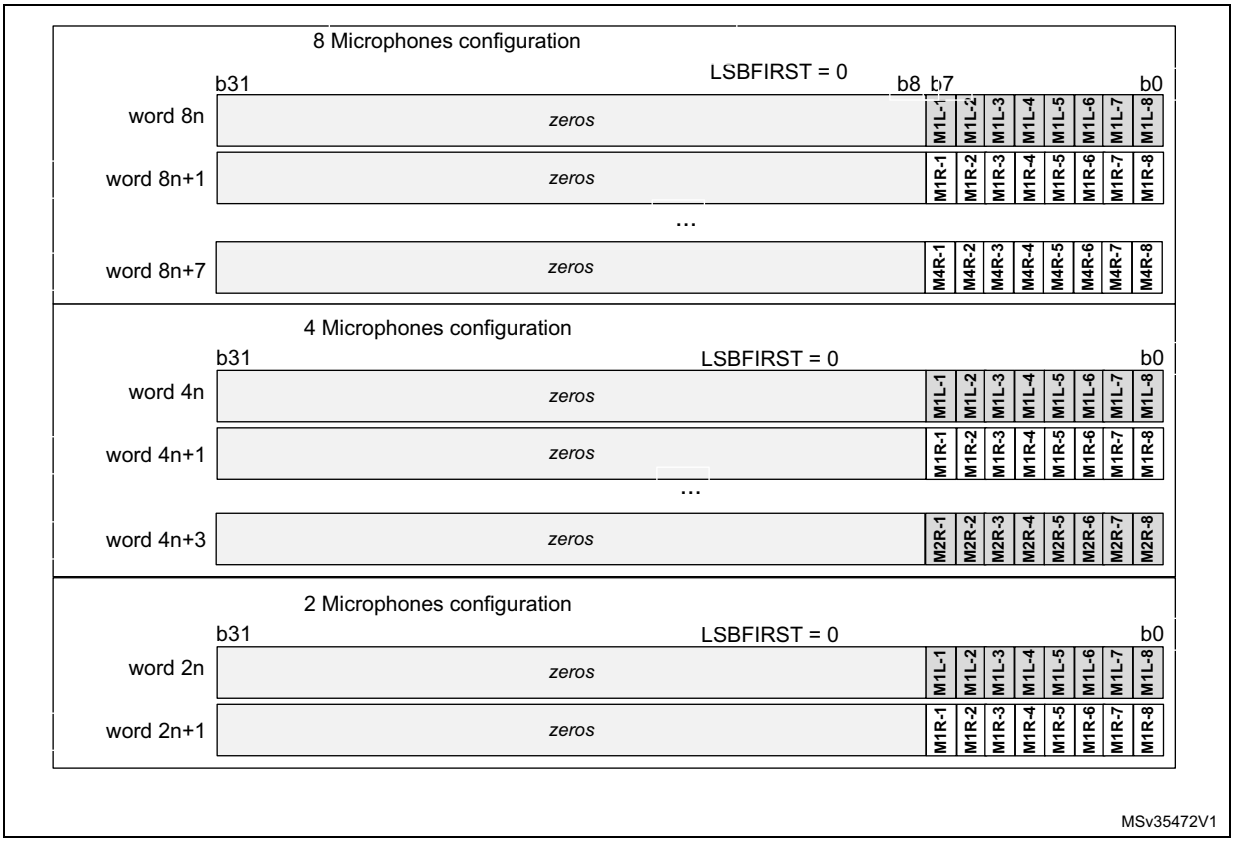


- **Using a 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 497*.

  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 497. SAI_ADR format in TDM, 8-bit slot width



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 362* to ensure a good connection with the PDM interface.

Table 362. 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 363*.
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 363*.
FSALL	0	Pulse width is one bit clock cycle
FSDEF	0	FS signal is a start of frame

Table 362. 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 363*.
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 363*.

Adjusting the bitstream clock rate

To properly program the SAI TDM interface, the user application must take into account the settings given in *Table 362*, 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 47.6.18*)

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 363. 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 362: 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.

47.4.11 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.

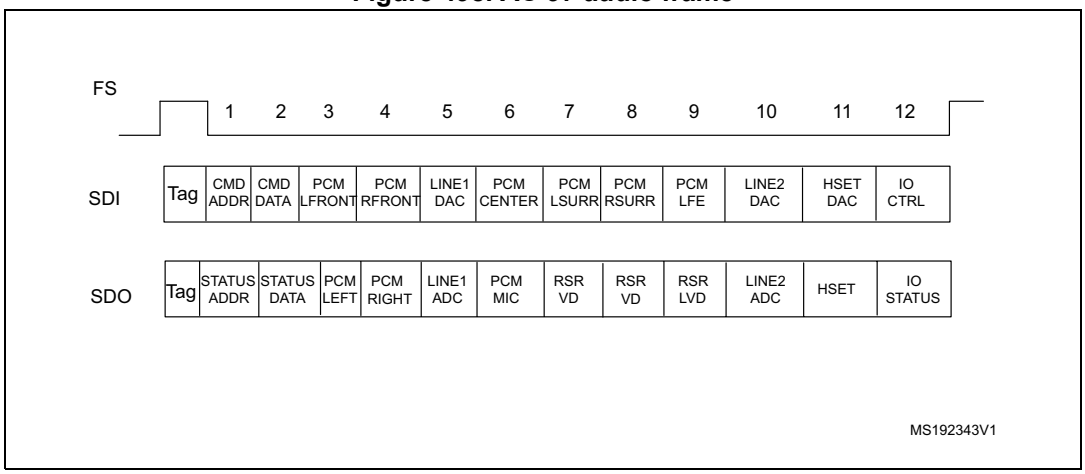
To select this protocol, set PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- 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 498 shows an AC’97 audio frame structure.

Figure 498. 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.

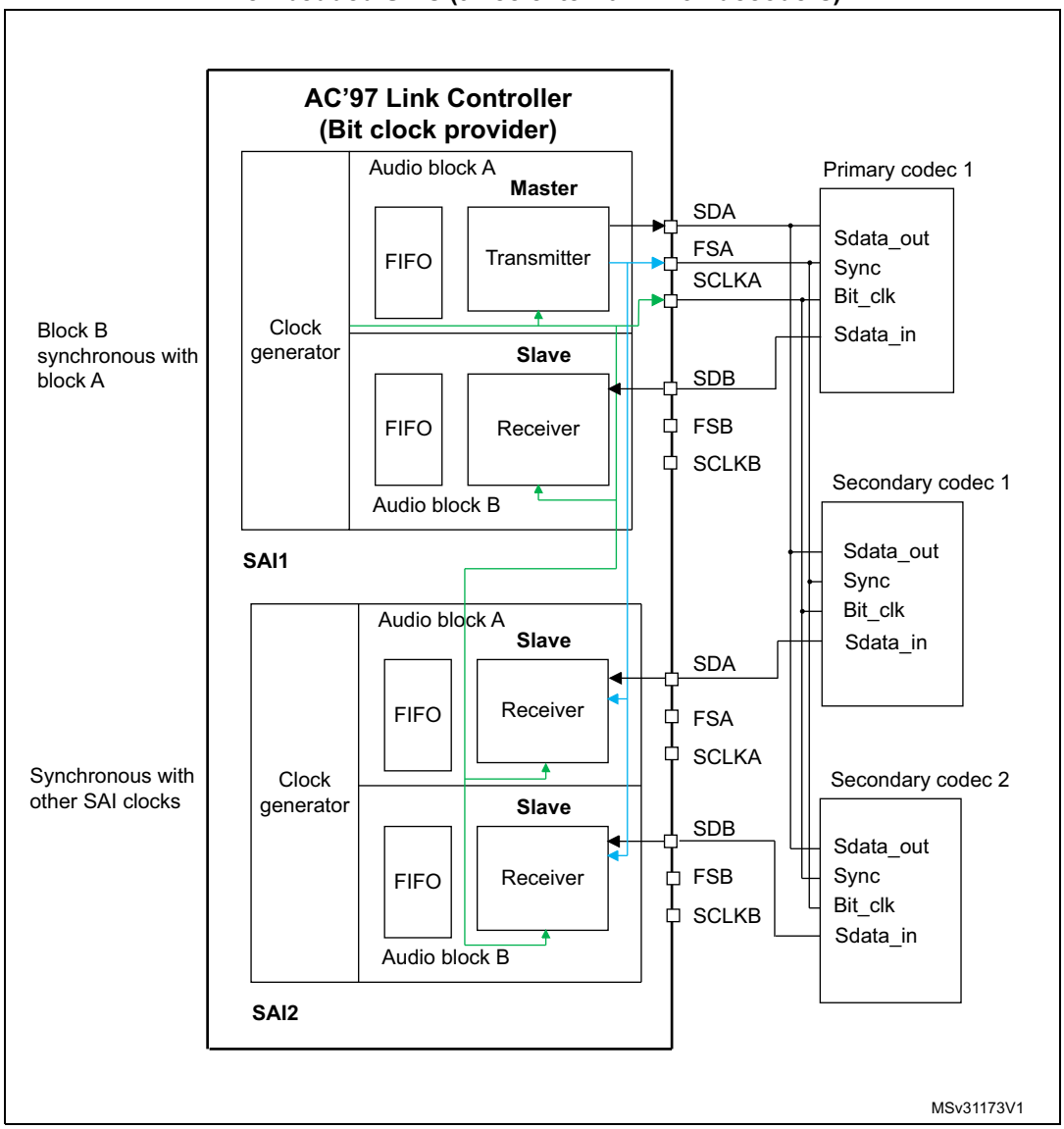
One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) allows controlling three external AC’97 decoders as illustrated in *Figure 499*.

In SAI1, the audio block A must be declared as asynchronous master transmitter whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.

Figure 499. Example of typical AC'97 configuration on devices featuring at least 2 embedded SAIs (three external AC'97 decoders)



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 47.4.8: SAI clock generator* must be used with FRL = 255, in order to generate the proper frame rate (F_{FS_x}).

47.4.12 SPDIF output

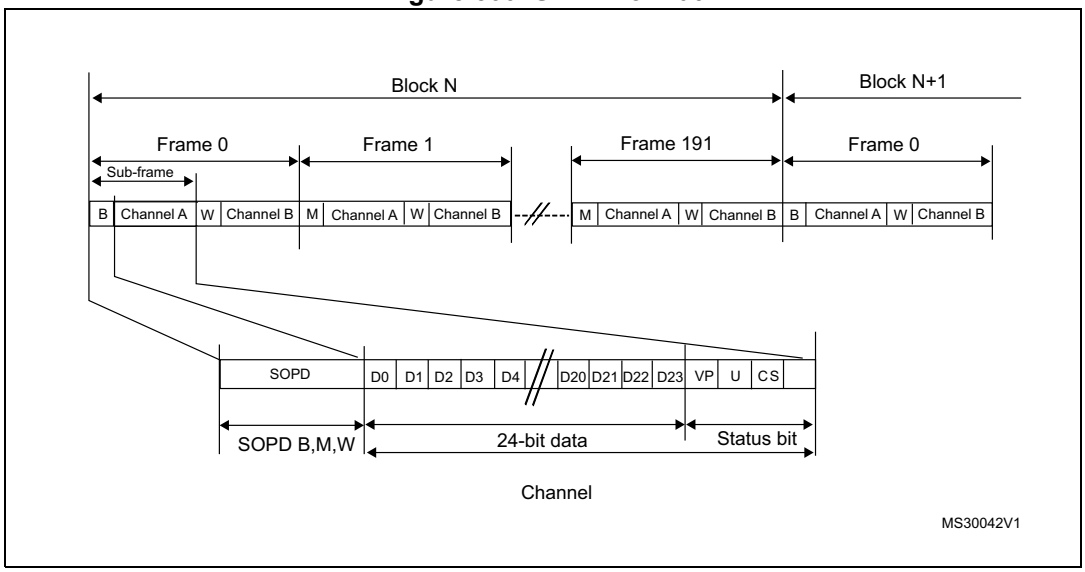
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

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 500*.

Figure 500. 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 364*). The next 28 bits of channel information are composed of 24 bits data + 4 status bits.

Table 364. 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:

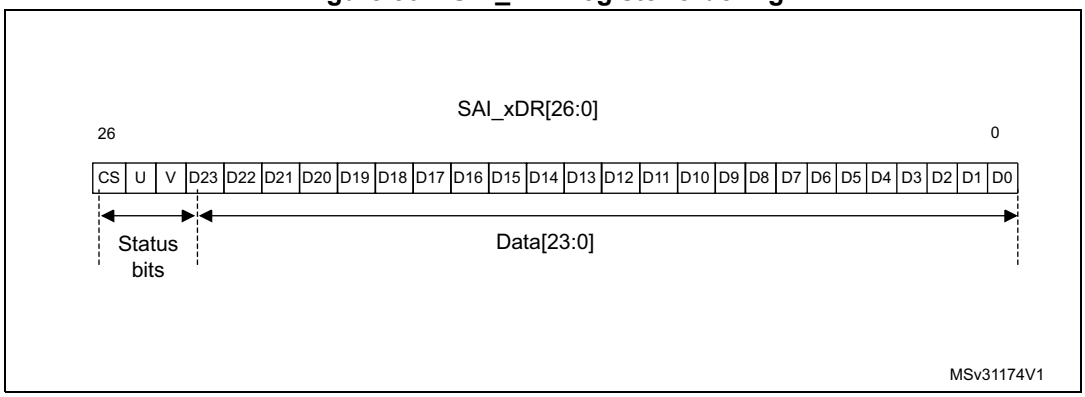
- SAI_xDR[26:24] contain the Channel status, User and Validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

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 501. 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 365*.

Table 365. 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 366. 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.

47.4.13 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 502*. 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).

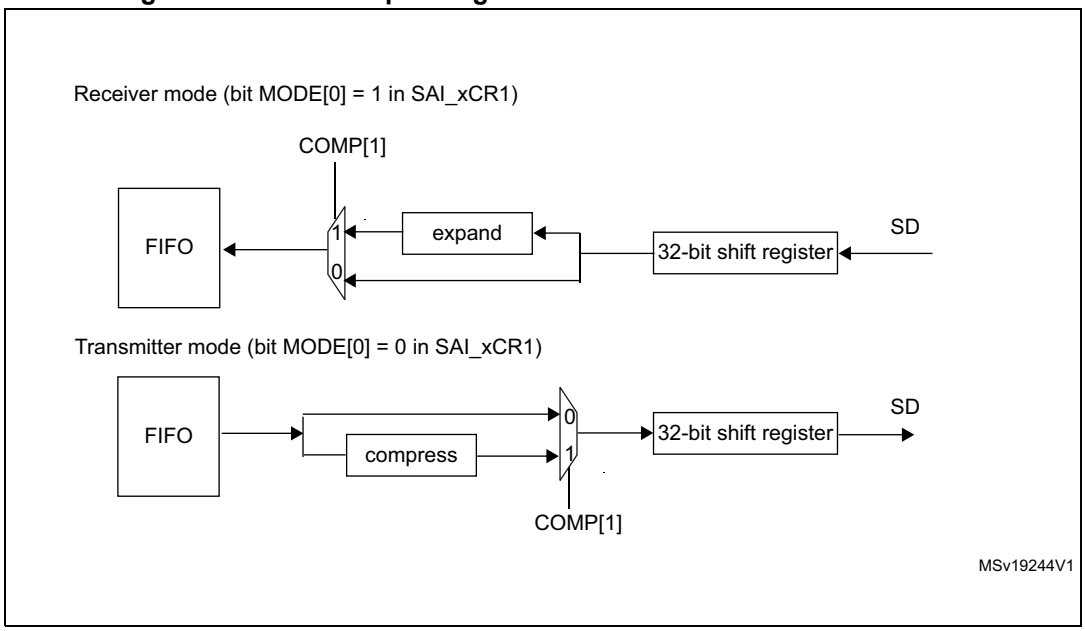
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

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 502. 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).

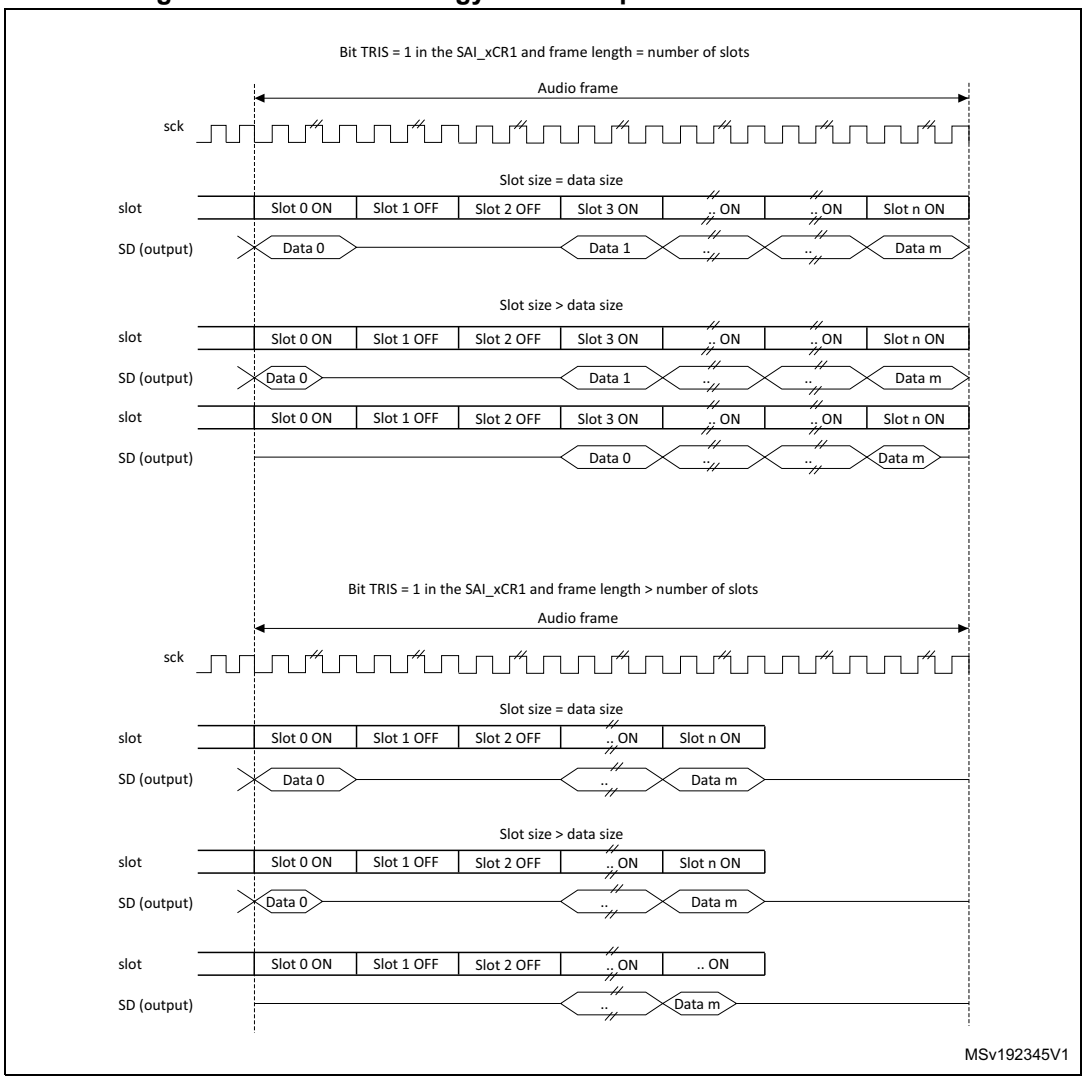
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in HI-z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

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.

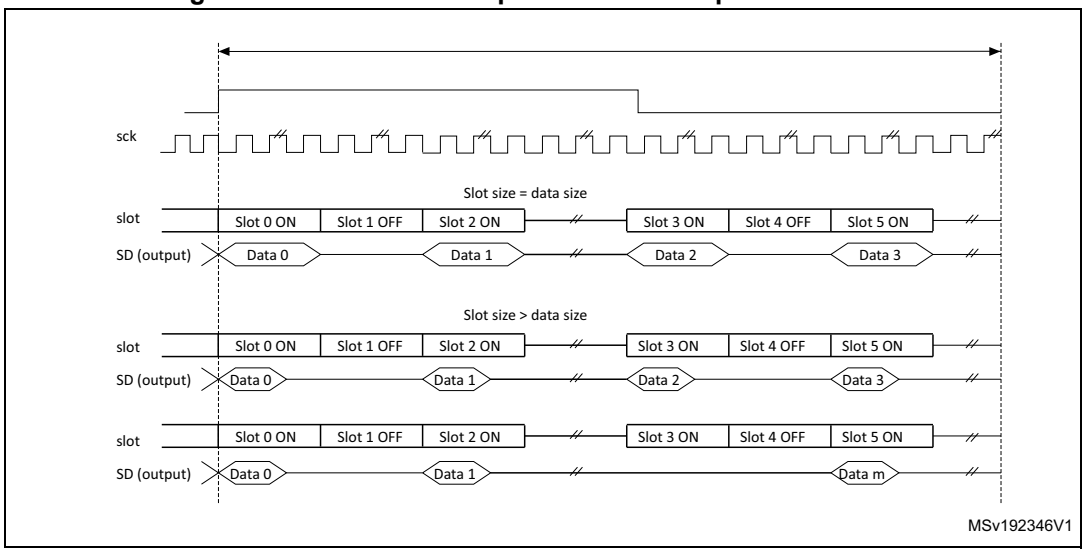
Figure 503 illustrates these behaviors.

Figure 503. 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 504* (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 504. 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 503* and *Figure 504* are replaced by a drive with a value of 0.

47.4.14 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.

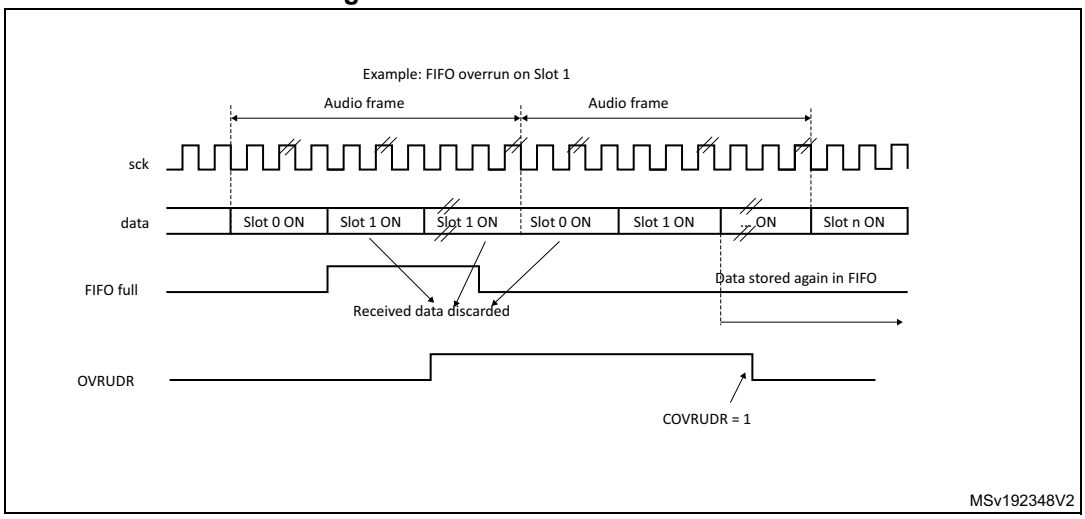
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

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 505*).

The OVRUDR flag is cleared when COVRUDR bit is set in the SAI_xCLRFR register.

Figure 505. Overrun detection error



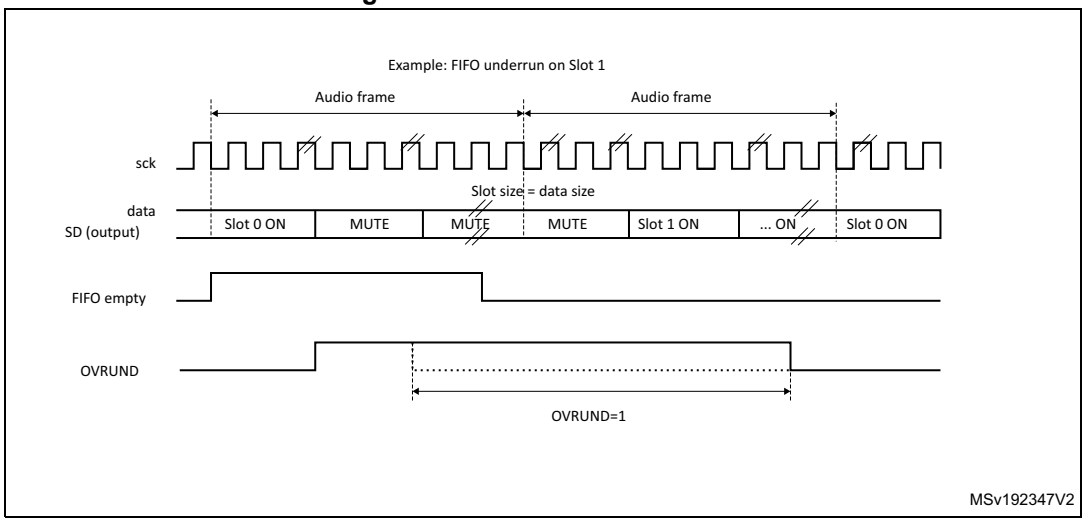
Underrun

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 506*). 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 506. 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.

47.4.15 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.

47.4.16 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 47.4.9: 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.

47.5 SAI interrupts

The SAI supports 7 interrupt sources as shown in *Table 367*.

Table 367. 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 47.4.9: 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.

47.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

47.6.1 SAI global configuration register (SAI_GCR)

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.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **SYNCOUT[1:0]**: Synchronization outputs

These bits are set and cleared by software.

00: No synchronization output signals. SYNCOUT[1:0] should be configured as “No synchronization output signals” when audio block is configured as SPDIF

01: Block A used for further synchronization for others SAI

10: Block B used for further synchronization for others SAI

11: Reserved. These bits must be set when both audio block (A and B) are disabled.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SYNCIN[1:0]**: Synchronization inputs

These bits are set and cleared by software.

Refer to *Table 359: External synchronization selection* for information on how to program this field.

These bits must be set when both audio blocks (A and B) are disabled.

They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCEN[1:0] = 10 in SAI_ACR1 or in SAI_BCR1 registers).

47.6.2 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 47.4.8: 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: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock should be configured in Slave mode.

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).

47.6.3 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 47.4.8: 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: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock should be configured in Slave mode.

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.

47.6.4 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

47.6.5 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

47.6.6 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.

47.6.7 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.

47.6.8 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.

47.6.9 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.

47.6.10 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.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									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.

47.6.11 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.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									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.

47.6.12 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 47.4.6: 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.

47.6.13 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 47.4.6: 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.

47.6.14 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.

47.6.15 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.

47.6.16 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.

47.6.17 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.

47.6.18 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.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
						rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

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

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 47.3: 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 47.3: 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 47.3: 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

47.6.19 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

47.6.20 SAI register map

The following table summarizes the SAI registers.

Table 368. 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	SAI_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
	Reset value																											0	0			0	0
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
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

Table 368. 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
0x0044	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																							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 87* for the register boundary addresses.

48 Secure digital input/output MultiMediaCard interface (SDMMC)

48.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (*e*•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and *e*•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.jedec.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 4.51*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit.
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 4.1.* (SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0.* Card support for two different databus modes: 1-bit (default) and 4-bit. (SPI mode and UHS-II mode not supported).
- Data transfer up to 104 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/*e*•MMC card at any one time and a stack of *e*•MMC.

48.2 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/*e*•MMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:

- Block mode: data block(s) with block size 2^N bytes with N in the range 0-14.
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- *e*•MMC Stream mode: continuous data stream

Data transfers to/from *e*•MMC cards are done in data blocks or streams.

Figure 507. SDMMC “no response” and “no data” operations

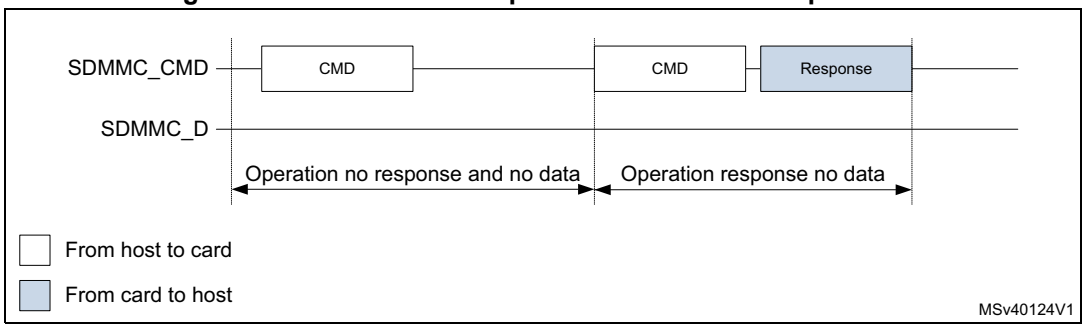
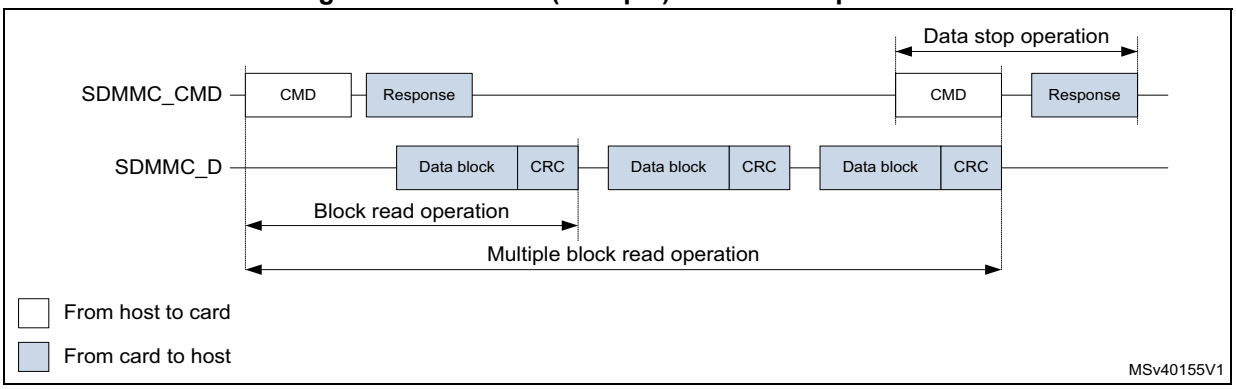
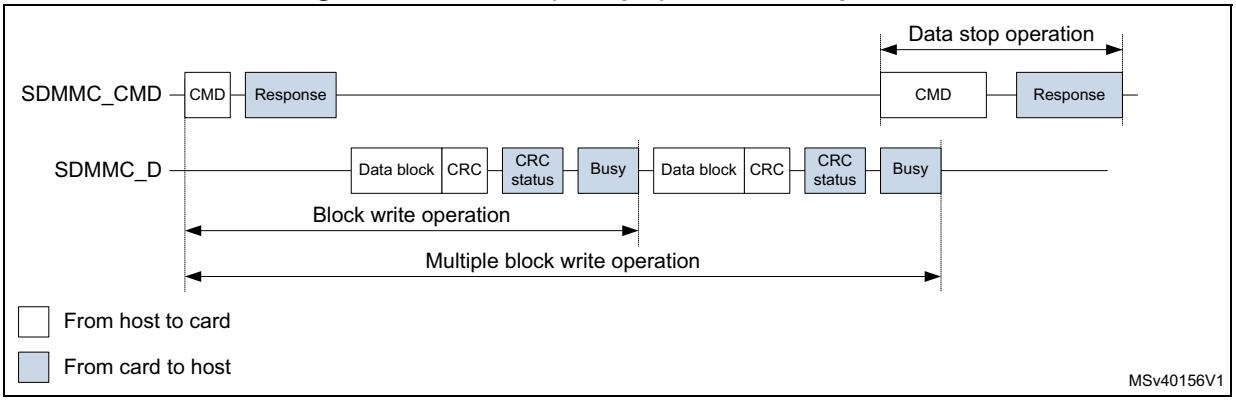


Figure 508. SDMMC (multiple) block read operation



*Note: The Stop Transmission command is not required at the end of a e•*MMC *multiple block read with predefined block count.*

Figure 509. SDMMC (multiple) block write operation



Note: The Stop Transmission command is not required at the end of an e•MMC multiple block write with predefined block count.

Note: The SDMMC does not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Figure 510. SDMMC (sequential) stream read operation

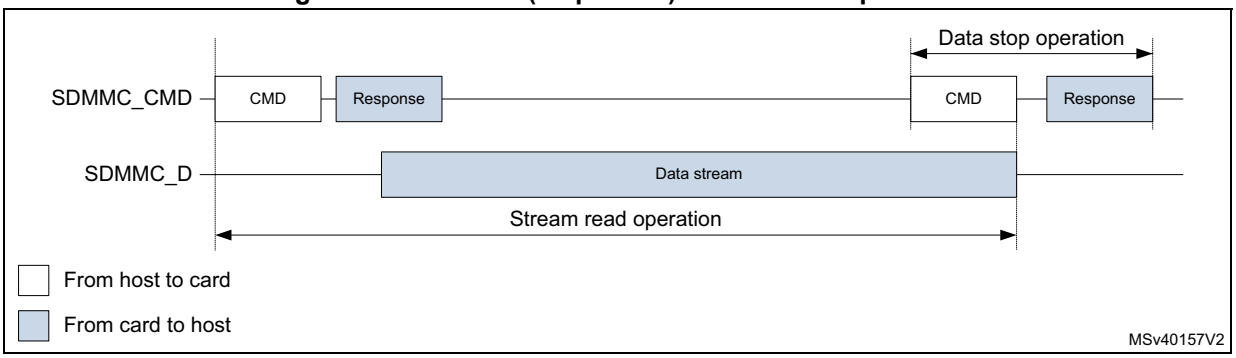
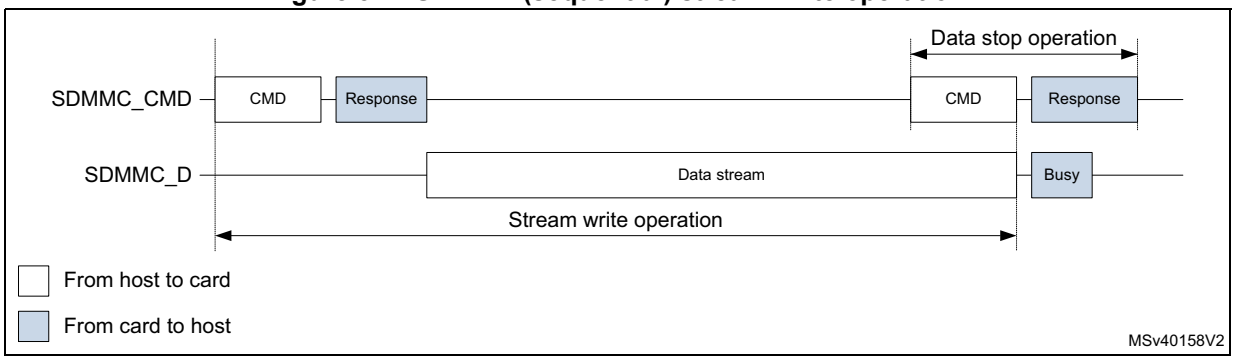


Figure 511. SDMMC (sequential) stream write operation



Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).

48.3 SDMMC operation modes

Table 369. SDMMC operation modes SD & SDIO

SDIO Bus Speed modes(1)(2)	Max Bus Speed (3) [Mbyte/s]	Max Clock frequency [MHz](4)	Signal Voltage [V]
DS (Default Speed)	12.5	25	3.3
HS (High Speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
DDR50	50	50	1.8
SDR50	50	100	1.8

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4bit bus width.
4. Maximum frequency depending on maximum allowed IO speed.

Table 370. SDMMC operation modes *e*•MMC

e•MMC bus speed modes (1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V]
Legacy compatible	26	26	3/1.8/1.2V
High speed SDR	52	52	3/1.8/1.2V
High speed DDR	104	52	3/1.8/1.2V

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. *e*•MMC bus speed with 8bit bus width.
4. Maximum frequency depending on maximum allowed IO speed.

48.4 SDMMC functional description

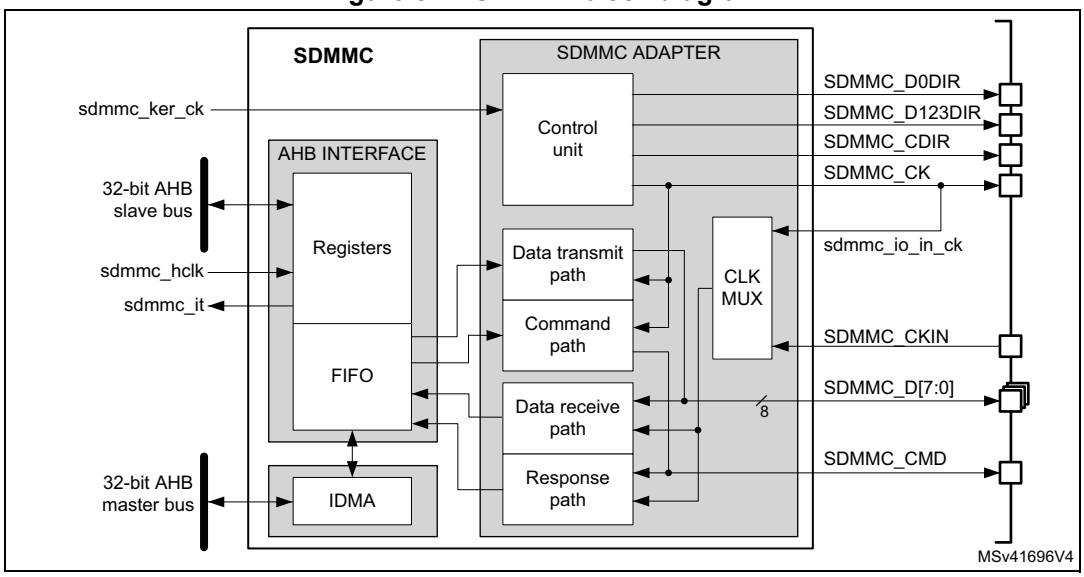
The SDMMC consists of three parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the *e*•MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.

48.4.1 SDMMC block diagram

Figure 512 shows the SDMMC block diagram.

Figure 512. SDMMC block diagram



48.4.2 SDMMC pins and internal signals

Table 371 lists the SDMMC internal input/output signals, *Table 372* the SDMMC pins (alternate functions).

Table 371. SDMMC internal input/output signals

Signal name	Signal type	Description
sdmmc_ker_ck	Digital input	SDMMC kernel clock
sdmmc_hclk	Digital input	AHB clock
sdmmc_it	Digital output	SDMMC global interrupt
sdmmc_io_in_ck	Digital input	SD/SDIO/*e*•MMC card feedback clock. This signal is internally connected to the SDMMC_CK pin (for DS and HS modes).

Table 372. SDMMC pins

Signal name	Signal type	Description
SDMMC_CK	Digital output	Clock to SD/SDIO/*e*•MMC card
SDMMC_CKIN	Digital input	Clock feedback from an external driver for SD/SDIO/*e*•MMC card. (for SDR12, SDR25, SDR50, DDR50)
SDMMC_CMD	Digital input/output	SD/SDIO/*e*•MMC card bidirectional command/response signal.
SDMMC_CDIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_CMD signal.
SDMMC_D[7:0]	Digital input/output	SD/SDIO/*e*•MMC card bidirectional data lines.
SDMMC_D0DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_D0 data line.
SDMMC_D123DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the data lines SDMMC_D[3:1].

48.4.3 General description

The **SDMMC_D[7:0]** lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an *e*•MMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The **SDMMC_D0DIR** signal indicates the I/O direction for the SDMMC_D0 data line, the **SDMMC_D123DIR** for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal **SDMMC_CDIR**.

SDMMC_CK clock to the card originates from **sdmmc_ker_ck**:

- When the sdmmc_ker_ck clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the sdmmc_ker_ck duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See *Figure 513*.

Figure 513. SDMMC Command and data phase relation

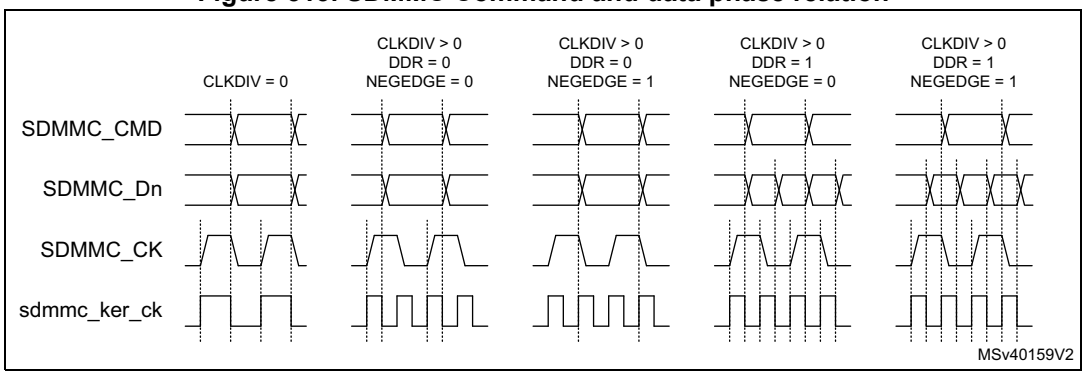


Table 373. SDMMC Command and data phase selection

CLKDIV	DDR	NEGEDGE	SDMMC_CK	Command out	Data out
0	x	x	= sdmmc_ker_ck	generated on sdmmc_ker_ck falling edge	
>0	0	0	generated on sdmmc_ker_ck rising edge	generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	
		1		generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	
	1	0		generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
		1		generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	

By default, the **sdmmc_io_in_ck** feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/*e*•MMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

48.4.4 SDMMC adapter

The SDMMC adapter (see *Figure 512: SDMMC block diagram*) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN.

Adapter register block

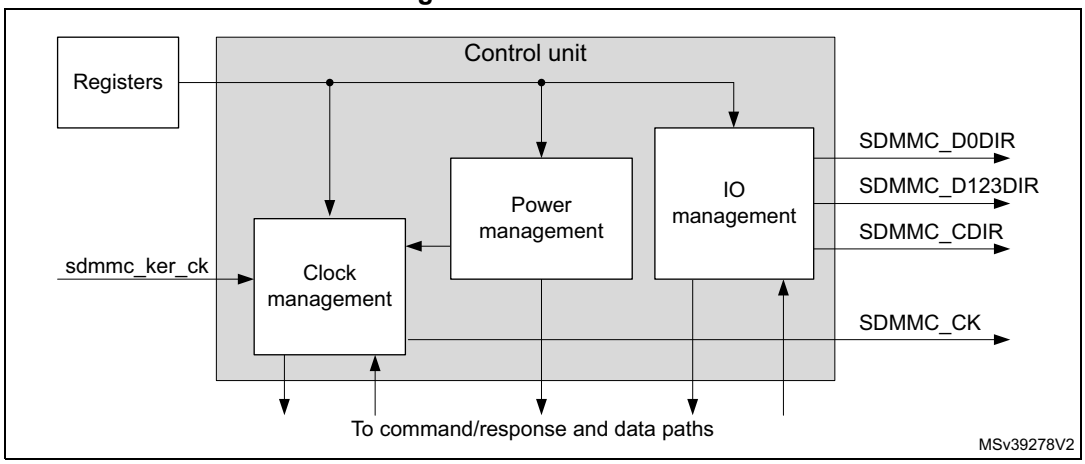
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in *Figure 514*, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 514. Control unit



The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:

- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for i.e. flow control.

The clock outputs are inactive:

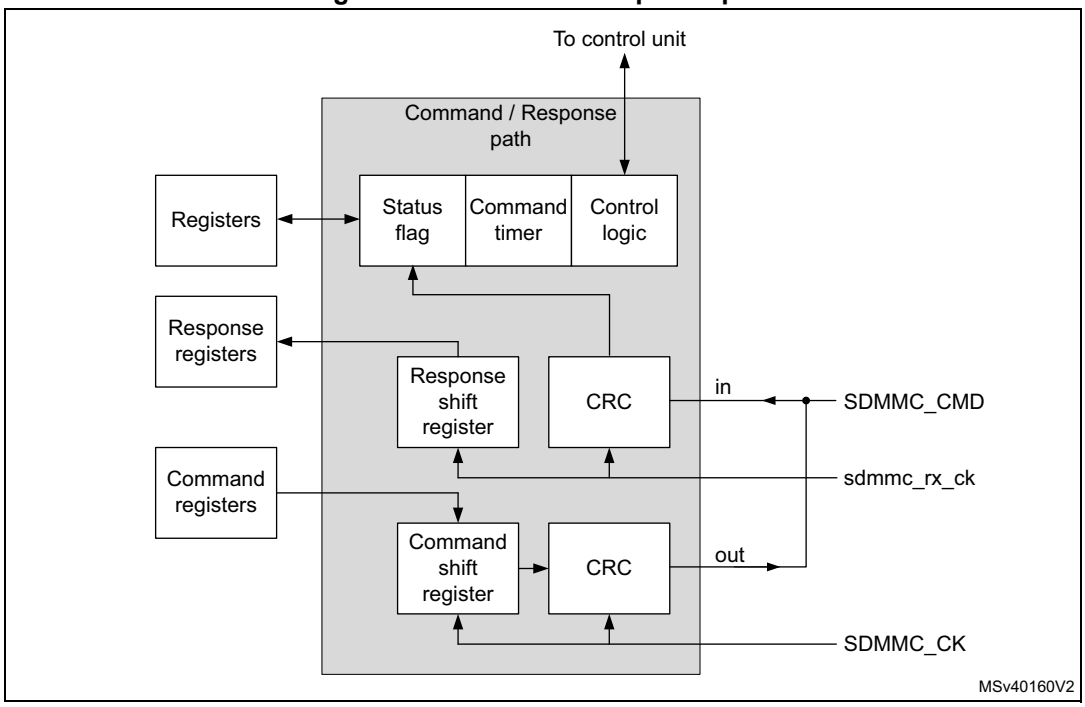
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock is restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

Command/response path

The command/response path subunit transfers commands and responses on the SDMMC_CMD line. The command path is clocked on the SDMMC_CK and sends commands to the card,. The response path is clocked on the sdmmc_rx_ck and receives responses from the card.

Figure 515. Command/response path

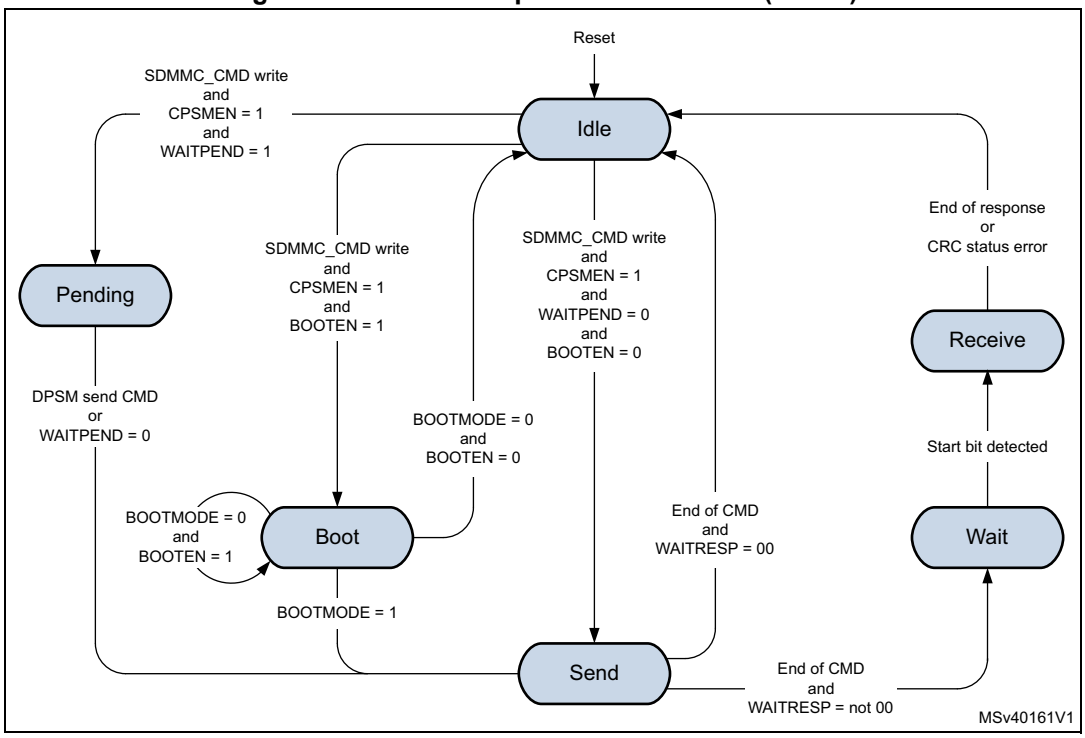


Command/response path state machine (CPSM)

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active, i.e. not in the Idle state, the CPSMACT bit is set.

Figure 516. Command path state machine (CPSM)



- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 & BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the Boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period is terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal is issued to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM moves to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 & BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command end bit, otherwise the

CMDSENT flag is generated immediately after the command end bit.
The RESPCMDR and RESPxR registers are not modified.

– If a command response is expected (WAITRESP = not 00) the CPSM moves to the Wait state and start the response timeout.

- **Wait**: The command path waits for a response.
  – When WAITINT bit is 0 the command timer starts running and the CPSM waits for a start bit.
  a) If a start bit is detected before the timeout the CPSM moves to the Receive state.
  b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are not modified.
  – When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (response start bit) from one of the cards.
  a) When a start bit is detected the CPSM moves to the Receive state.
  b) When writing WAITINT to 0 (interrupt mode abort), the host sends a response by its self and on detecting the start bit the CPSM move to the Receive state.
- **Receive**: The command response is received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present is verified against the internally generated CRC code.
  – When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period is started after the response.
  When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there is no interrupt period started.
  – When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period is terminated after the response.
  – When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
  - When BOOTMODE = 1 & BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response end bit, otherwise the CMDREND flag is generated immediately after the response end bit.
  - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal is issued to the DPSM at the end of the command response.
  – When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
- **Pending**: According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  – When DATALENGTH =< 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  – When DATALENGTH > 5 bytes, the CPSM DataEnable signal is issued to the DPSM to start the data transfer. The CPSM waits for a send CMD signal from the

DPSM before moving to the Send state. This enables i.e. the CMD12 Stop Transmission command to be sent aligned with the data.

– When writing WAITPEND to 0, the CPSM moves to the Send state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the Boot state, and when:
  – BOOTMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal is issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  – BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

Note: The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

Note: The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for *e*•MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and *e*•MMC cards is shown in *Table 374.*

The command token data is taken from 2 registers, one containing a 32-bits argument and the other containing the 6-bits command index (six bits sent to a card).

Table 374. Command token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	x	Command index
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Next to the command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various

coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and *e*•MMC cards are shown in *Table 375*, *Table 376* and *Table 377*.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

The response token data is stored in 5 registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

Table 375. Short response with CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Table 376. Short response without CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	1111111	(reserved 1111111)
0	1	1	End bit

Table 377. Long response with CRC token format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127:8	x	CID or CSD slices
	7:1	x	CRC7 (included in CID or CSD)
0	1	1	End bit

The command/response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the

SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see *Figure 513*.

The command and short response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmission bit, command index, and command argument (or card status).

For the long response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmission bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

CRC[6:0] = remainder $[(M(x) * x^7) / G(x)]$

$G(x) = x^7 + x^3 + 1$

$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit before CRC}) * x^0$

Where n = 39 or 119.

The CPSM allows to send a number of specific commands to handle various operating modes when CPSMEN is set, see *Table 378*.

Table 378. Specific Commands overview

VSWITCH	BOOTEN	BOOTMOD	CMDTRAN	WAITPEND	CMDSTOP	WAITINT	Description
1	x	x	x	x	x	x	Start Voltage Switch Sequence
0	1	x	x	x	x	x	Start normal boot
0	1	1	x	x	x	x	Start alternative boot
0	0	1	x	x	x	x	Stop alternative boot.
0	0	0	1	x	x	x	Send command with associated data transfer.
0	0	0	0	1	1	x	*e*•MMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	1	0	x	*e*•MMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	0	1	x	Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.
0	0	0	0	0	0	1	Enter *e*•MMC wait interrupt (Wait-IRQ) mode.
0	0	0	0	0	0	0	Any other none specific command

The command/response path implements the status flags and associated clear bits shown in *Table 379*:

Table 379. Command path status flags

Flag	Description
CMDSENT	Set at the end of the command without response. (CPSM moves from Send to Idle)
CMDREND	Set at the end of the command response when the CRC is OK. (CPSM moves from Receive to Idle)
CCRCFAIL	Set at the end of the command response when the CRC is FAIL. (CPSM moves from Receive to Idle)
CTIMEOUT	Set after the command when no response start bit received before the timeout. (CPSM moves from Wait to Idle)
CKSTOP	Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped. (no impact on CPSM)
VSWEND	Set after the voltage switch (VSWITCH = 1) timeout of 5 ms + 1 ms. (no impact on CPSM)
CPSMACT	Command transfer in progress. (CPSM not in Idle state)

The command path error handling is shown in *Table 380*:

Table 380. Command path error handling

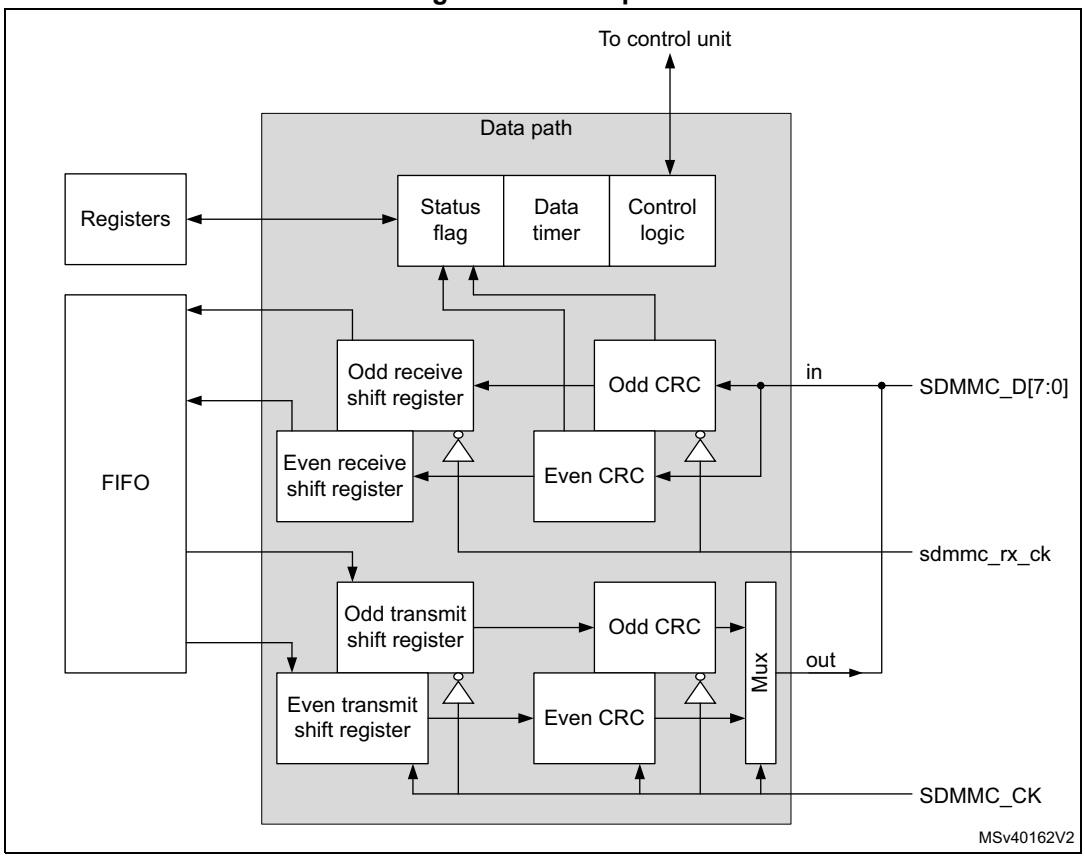
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card(1)	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command(1)	Move to Idle
		Transmission error	Command accepted	Resend command(1)	

1. When CMDTRANS is set, also a stop_transmission command must be send to move the DPSM to Idle.

Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. *Figure 517* shows the data path block diagram.

Figure 517. Data path



The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)

In DDR mode, data is sampled on both edges of the SDMMC_CK according the following rules, see also *Figure 518* and *Figure 519*:

- On the rising edge of the clock odd bytes are sampled.
- On the falling edge of the clock even bytes are sampled.
- Data payload size is always a multiple of 2 Bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, end bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode the SDMMC_CK clock division must be >= 2.

Figure 518. DDR mode data packet clocking

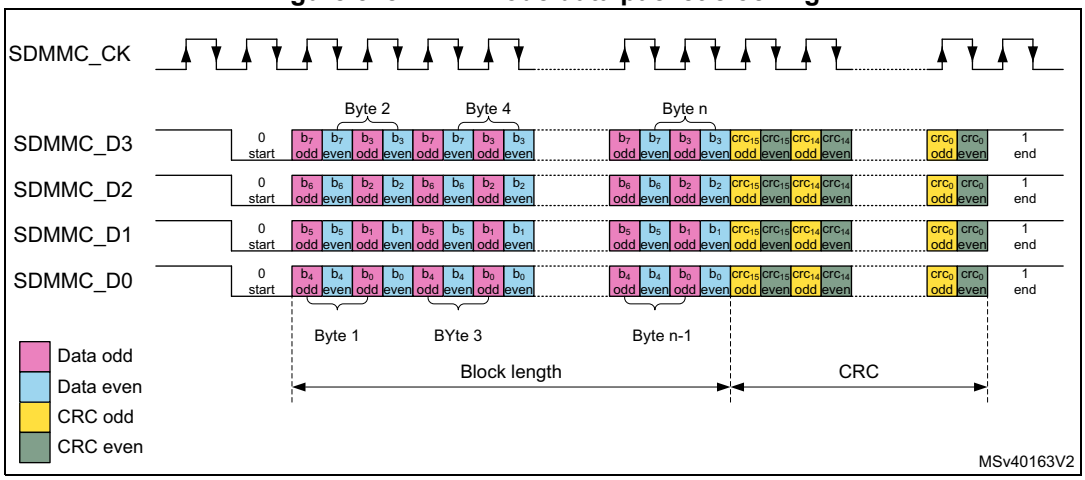
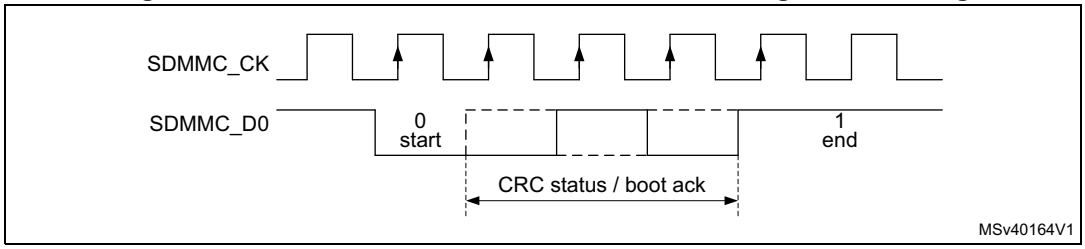


Figure 519. DDR mode CRC status / boot acknowledgment clocking



Data path state machine (DPSM)

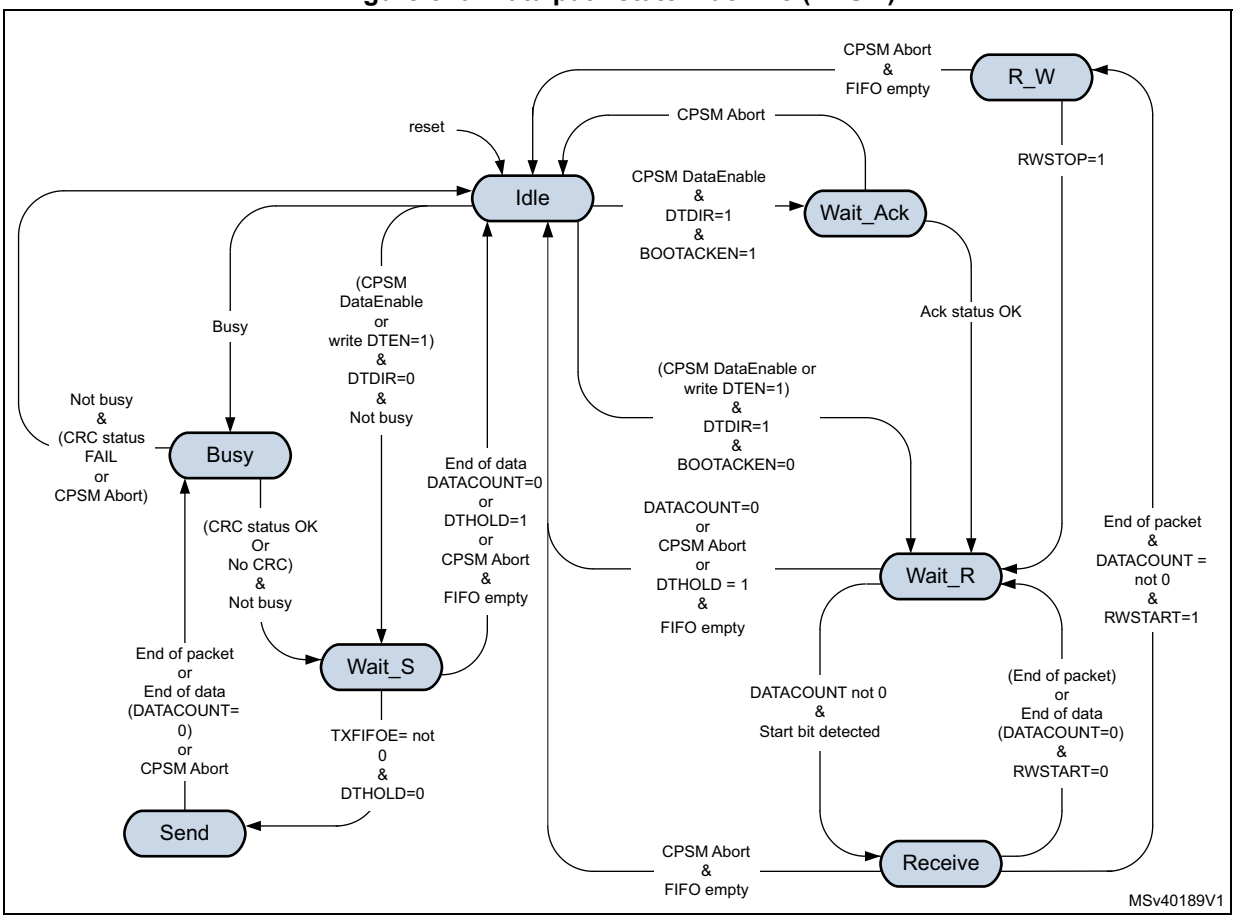
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in *Figure 520*. When ever the DPSM is active, i.e. not in the Idle state, the DPSMACT bit is set.

Figure 520. Data path state machine (DPSM)



- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM is activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:

  – When the data direction bit (DTDIR) indicates send, moves to the Wait_S.

  – When the data direction bit (DTDIR) indicates receive, moves to the
  - Wait_R when BOOTACKEN register bit is clear.
  - Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM keeps the SDMMC_CK clock active and move to the Busy state.

Note: DTEN must not be used to start data transfer with SD, SDIO and e•MMC cards.

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.
- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the Wait_R state.
  - If the CPSM Abort signal is set:
    If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    If DATACOUNT is zero normal operation is continued, there is no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When holding the timeout is disabled. When an CPSM Abort signal is received during holding, the transfer is aborted.

- When DATACOUNT = 0, the transfer is completed normally and there is no DHOLD flag.

– When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **R_W** state: the data path Read Wait the bus.
  – The DPSM moves to the Wait_R state when the Read Wait stop bit (RWSTOP) is set, and start the receive timeout.
  – If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.
- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  – In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  – In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.
  a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
  - R_W state when RWSTART = 1 and DATACOUNT > zero, the DBCKEND flag is set.
  - Wait_R state otherwise.
  b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
  - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
  - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  – In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
  – When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
  – When an CPSM Abort signal is received:
  - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
  - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT=0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.
  – When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the Send state.
  - If the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
    - When DTHOLD is disabled, the DATAEND flag is set.
    - When DTHOLD is enabled, the DHOLD flag is set.
  - When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM moves to the Idle state and the DHOLD flag is set.
    - When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM moves to the Idle state and issues the DHOLD flag.
  - When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT=0, normal operation is continued, there is no DABORT flag since the transfer has completed normally.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
    Before sending the last stream Byte according to DATACOUNT, the DPSM issues a trigger on the send CMD signal. This signal is used by the CPSM to sent any pending command. (i.e. CMD12 Stop Transmission command)
  - If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT=0, the transfer is aborted. The DPSM sends a last data bit followed by an end bit. The FIFO is disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer

with DATACOUNT=0, the transfer is completed normally, there is no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate “non-erroneous transmission” or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it remains in the Busy state.
  - If a CRC status token is expected and indicates “erroneous transmission”.
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO is disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT=0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.

The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0,

- In receive the start bit must be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

Table 381. Data token format

Description	Start bit	Data(1)	CRC16	End bit	DTMODE
Block data	0	(DBLOCKSIZE, DATALENGTH)	yes	1	00
SDIO multibyte	0	(DATALENGTH)	yes	1	01
e•MMC stream	0	(DATALENGTH)	no	1	10

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in *Table 382*:

Table 382. Data path status flags and clear bits

Flag		Description
DATAEND	TX	Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from Wait_S to Idle)
	RX	Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from Wait_R to Idle)
	Boot	

Table 382. Data path status flags and clear bits (continued)

Flag		Description
DCRCFAIL	TX	Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in Busy when there is still data to send and wait for CPSM Abort) (DPSM moves from Busy to Idle when all data has been sent) or DPSM has been started with DTEN
	RX	Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in Receive when there is still data to be received and wait for CPSM Abort) (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)
	Boot	
ACKFAIL	Boot	Set at the end of the boot acknowledgment when fail. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DTIMEOUT	CMD R1b	Set after the command response no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort)
	TX	Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.
	RX	Set when no start bit received before the timeout. (DPSM stays in Wait_R and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)
	Boot	
ACKTIMEOUT	Boot	Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DBCKEND	TX	When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT >0). (DPSM moves from Busy to Wait_S)
	RX	When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from Receive to R_W)
	Boot	
DHOLD	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from Wait_S to Idle)
	RX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT >0). (DPSM moves from Wait_R to Idle)
DABORT	CMD R1b	When CPSM Abort event has been sent by the CPSM and busy has finished. (DPSM moves from Busy to Idle)
	TX	When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)
	RX	
	Boot	
BUSYD0END	CMD R1b	Set after the command response when end of busy before the timeout. (DPSM moves from Busy to Idle)
DPSMACT		Data transfer in progress. (DPSM not in Idle state)

The data path error handling is shown in *Table 383*:

Table 383. Data path error handling

Error	DPSM state	Cause	Card action	Host action	DPSM action
Timeout	Wait_Ack	No Ack in time	unknown	Card cycle power	Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)
	Wait_R	No start bit in time	unknown	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
			unknown	Stop boot procedure	
	Busy	Busy too long (due to data transfer)	unknown	Stop data reception Send stop transmission command	
		Busy too long (due to R1b)	unknown	Send reset command	
CRC	Receive	transmission error	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
CRC status	Busy	Negative status	Ignore further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle
		transmission error	wait for further data		
Ack status	Wait_Ack	transmission error	Send boot data	Stop boot procedure	On CPSM Abort move to Idle
Overrun	Receive	FIFO full	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
Underrun	Send	FIFO empty	Receive further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle

Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.

The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see *Table 384*.

Table 384. Data FIFO access

Data FIFO access	IDMAEN
From firmware via AHB slave interface	0
From IDMA via AHB master interface	1

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   - For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
   - Configures the FIFO in transmit mode.
3. Enable the data transfer
   - either by sending a command from the CPSM with the CMDTRANS bit set
   - or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
   - The DPSM stays in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.

- The SDMMC keeps sending data as long as FIFO is not empty, hardware flow control during data transfer is used to prevent FIFO underrun.

5. Write data to the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. When last data has been written wait for end of data (DATAEND flag)
   – SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, firmware must stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in *Table 385*.

Table 385. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when half or more transmit FIFO words are empty.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface.When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses.

The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
3. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
   – The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
5. Read data from the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   - When last data has been received end of data (DATAEND flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, the firmware must read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, the firmware must stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in *Table 386*.

Table 386. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when half or more receive FIFO words contain valid data.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

CLKMUX unit

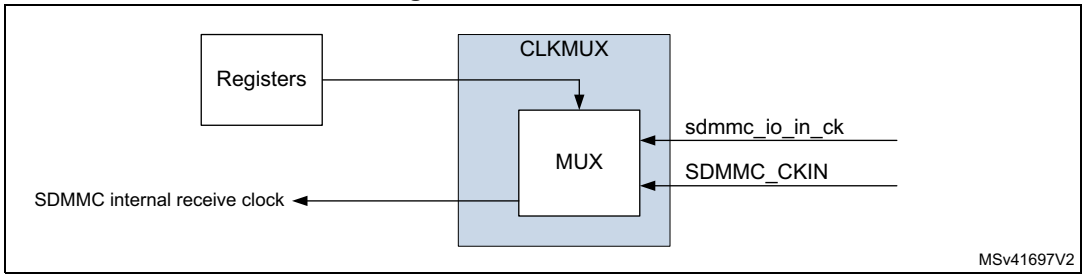
The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.

The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.

The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

Figure 521. CLKMUX unit



The sdmmc_rx_ck source must be changed when the CPSM and DPSM are in the Idle state.

48.4.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

SDMMC FIFO

The FIFO access is restricted to word access only:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.
- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

48.4.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.
- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- double buffered channel

Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE0. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Double buffered channel

In double buffer configuration the data at the memory side is subsequently accessed from 2 buffers, one located from base address IDMABASE0 and a second located from base address IDMABASE1. This allows firmware to process one memory buffer while the IDMA is accessing the other memory buffer. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. It is possible to update the base address of the buffers on-the-fly when the channel is enabled, the following rule apply:

- When IDMABACT bit is ‘0’ the IDMA hardware uses the IDMABASE0 to access memory. When attempting to write to this register by Firmware the write is discarded, IDMABASE0 data isl not changed. Firmware is allowed to write IDMABASE1.
- When IDMABACT bit is ‘1’ the IDMA hardware uses the IDMABASE1 to access memory. When attempting to write to this register by Firmware the write is discarded, IDMABASE1 data is not changed. Firmware is allowed to write IDMABASE0.

When the IDMA has finished transferring the data of one buffer the buffer transfer complete flag (IDMABTC) is set and the IDMABACT bit toggles where after the IDMA continues

transferring data from the other buffer. When the IDMA has finished transferring all data and the DPSM has completed the transfer the DATAEND flag is set.

The IDMABASEn address must be word aligned.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space.

On a IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set. Depending when the IDMA transfer error occurs, it normally causes the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND).
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.

The IDMATE is generated on an other SDMMC transfer interrupt (TXUNDERR. RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).

48.4.7 AHB and SDMMC_CK clock relation

The AHB must at least have 3x more bandwidth than the SDMMC bus bandwidth i.e. for SDR50 4-bit mode (50 Mbyte/s) the minimum sdmmc_hclk frequency is 37.5 MHz (150 Mbyte/s).

Table 387. AHB and SDMMC_CK clock frequency relation

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
e•MMC DS	8	26	19.5
e•MMC HS	8	52	39
e•MMC DDR52	8	52	78
SD DS / SDR12	4	25	9.4
SD HS / SDR25	4	50	18.8
SD DDR50	4	50	37.5
SD SDR50	4	100	37.5

48.5 Card functional description

48.5.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO Read Wait operation by stopping the clock
- SDIO Read Wait operation by SDMMC_D2 signaling

Table 388. SDIO special operation control

Operation mode	SDIOEN	RWMOD	RWSTOP	RWSTART	DTDIR
Interrupt detection	1	X	X	X	X
Suspend/Resume operation	X	X	X	X	X
Read Wait SDMMC_CK clock stop (START)	X	1	0	1	1
Read Wait SDMMC_CK clock stop (STOP)	X	1	1	1	1
Read Wait SDMMC_D2 signaling (START)	X	0	0	1	1
Read Wait SDMMC_D2 signaling (STOP)	X	0	1	1	1

SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

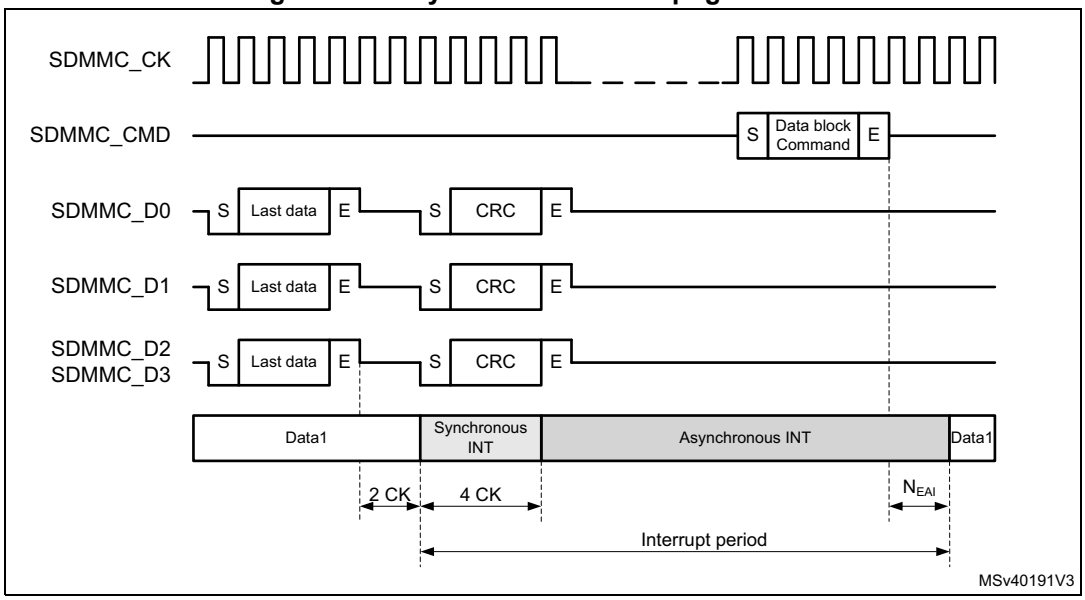
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in *Table 388*.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

Figure 522. Asynchronous interrupt generation



The timing of the interrupt period is depended on the bus speed mode:

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 523. Synchronous interrupt period data read

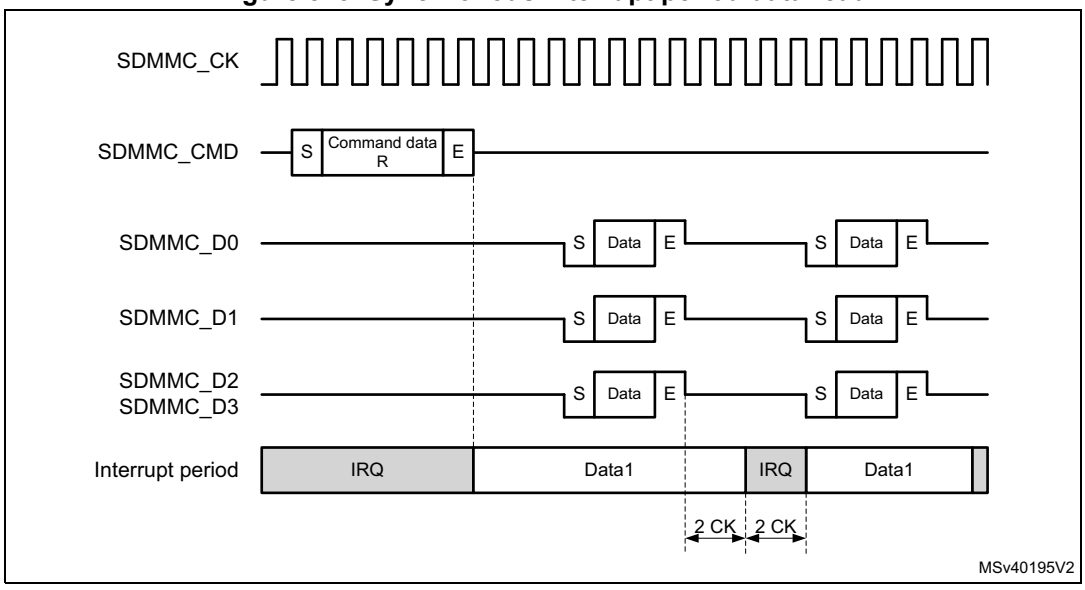
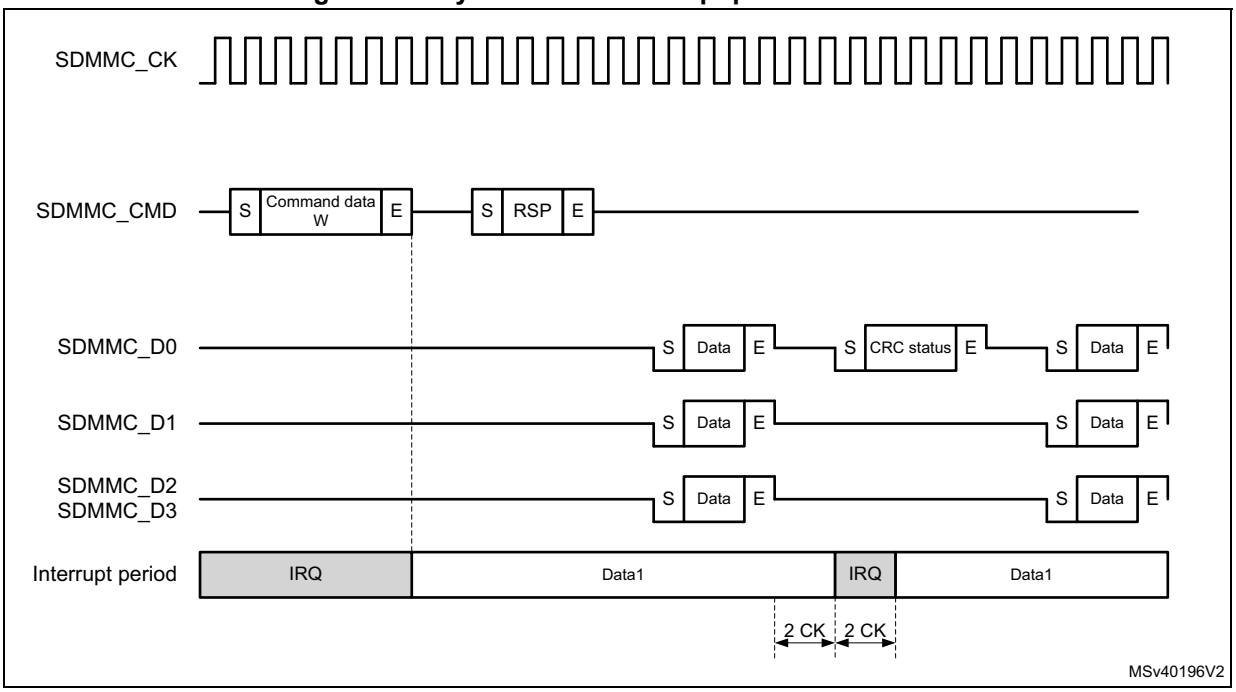


Figure 524. Synchronous interrupt period data write



In SDR50 and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the end bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command end bit are not detected by the host during this interrupt period.
- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host resumes the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 525. Asynchronous interrupt period data read

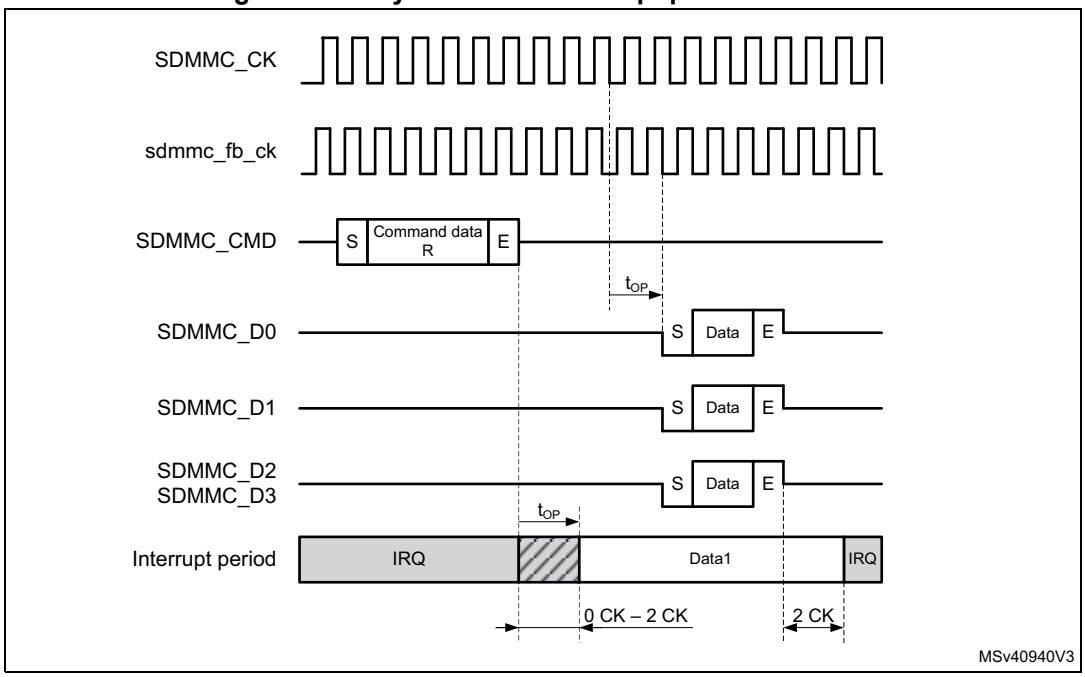
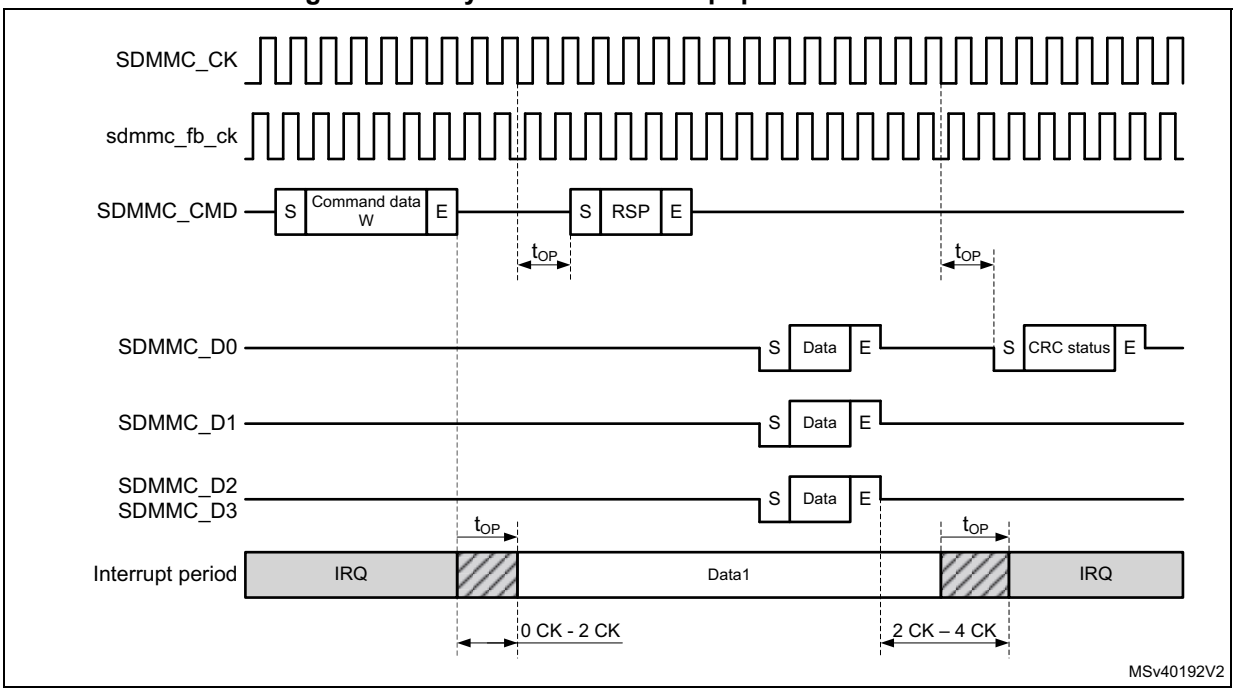


Figure 526. Asynchronous interrupt period data write



When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC masks the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according *Table 389*.

Table 389. 4-bit mode Start, interrupt, and CRC-status Signaling detection

SDMMC data line	Start	Interrupt	CRC-status
SDMMC_D0	0	1 or CRC-status	0
SDMMC_D1	0	0	X
SDMMC_D2	0	1 or Read Wait	X
SDMMC_D3	0	1	X

SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the *e*•MMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC_D[3:0] line(s)
2. Requests the lower-priority or slower transaction to suspend
3. Waits for the transaction suspension to complete
4. Begins the higher-priority transaction
5. Waits for the completion of the higher priority transaction
6. Restores the suspended transaction

The card receiving a suspend command responds with its current bus status. Only when the bus has been suspended by the card the bus status indicates suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted. (data transfer has suspended)
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during Read Wait.

For the host to know if the bus has been released it must check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation must be saved where after an other operation can start.

The suspend command must be sent with the CMDSUSPEND bit set. This allows to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transfered when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps must be taken by firmware:

1. The normal receive process must be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO must be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO must be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation must be determined from the remaining number of bytes indicated by the DATACOUNT.

Note: When a DTIMEOUT flag occurs during the suspend procedure, this must be ignored.

To resume receiving data from the card, the following steps must be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) must be programmed in DATALENGTH.
2. The DPSM must be configured to receive data in the DTDIR bit.
3. The resume command must be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which ends the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card resumes sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps must be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM must be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO must be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming must be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command must be sent by the CPSM with the CMDSUSPEND bit set. This allows to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps must be taken by firmware:

1. The remaining number of data bytes must be programmed in DATALENGTH.
2. The DPSM must be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This ends the interrupt period and start the data transfer. The

DPSM either goes to the Wait_S state when SDMMC_D0 does not signal busy, or goes to the Busy state when busy is signaled.

3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transfered.
4. When IDMAEN = 0: Firmware must start filling the FIFO with the remaining data.

SD I/O Read Wait

There are 2 methods to pause the data transfer during the Block gap:

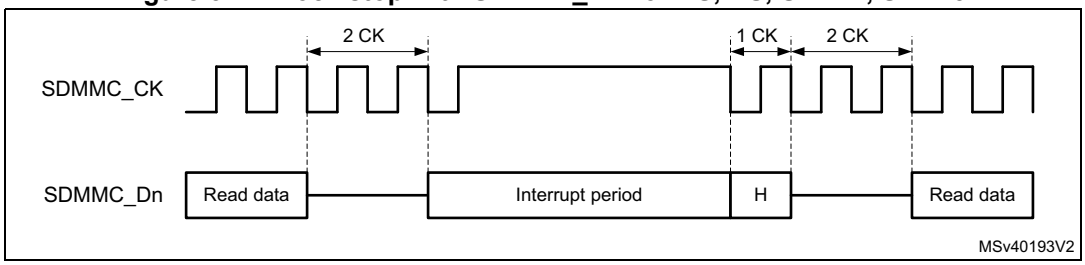
1. Stopping the SDMMC_CK.
2. Using Read Wait signaling on SDMMC_D2.

The SDMMC can perform a Read Wait with register settings according *Table 388*.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, DDR) each method has a different characteristic.

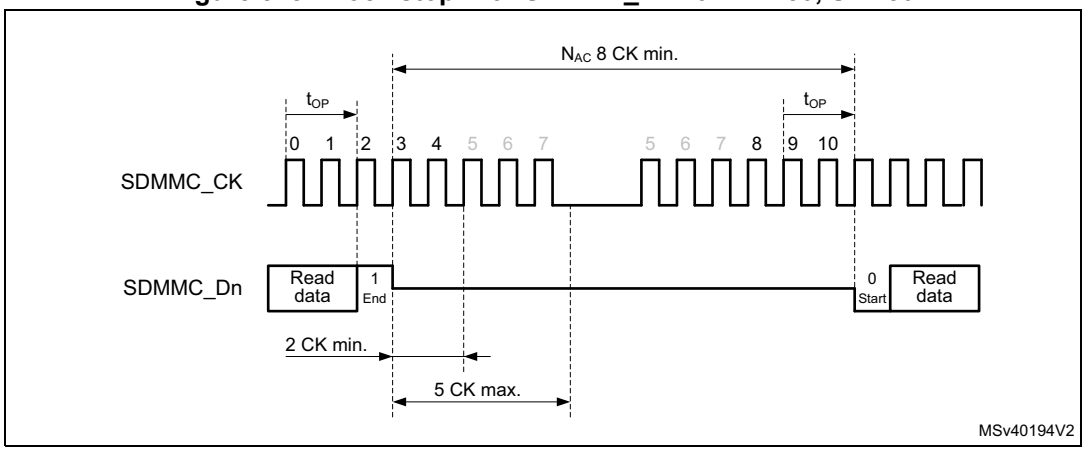
The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the end bit. When ready the host resumes by restarting clock, see *Figure 527*.

Figure 527. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC_CK for SDR50 and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the end bit. When ready the host resumes by restarting clock, see *Figure 528*. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

Figure 528. Clock stop with SDMMC_CK for DDR50, SDR50



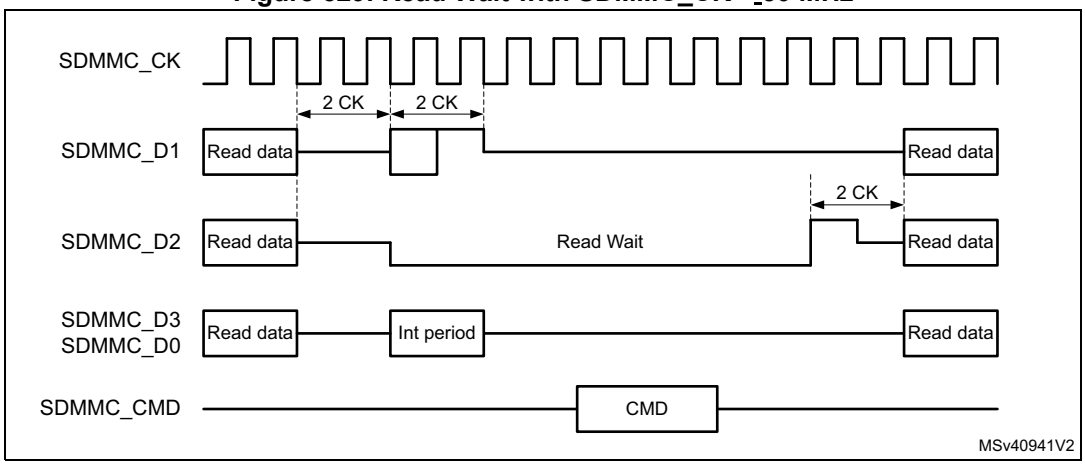
In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a Read Wait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional Read Wait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation allows the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the Read Wait protocol, the host must test capability bits in the internal card registers.

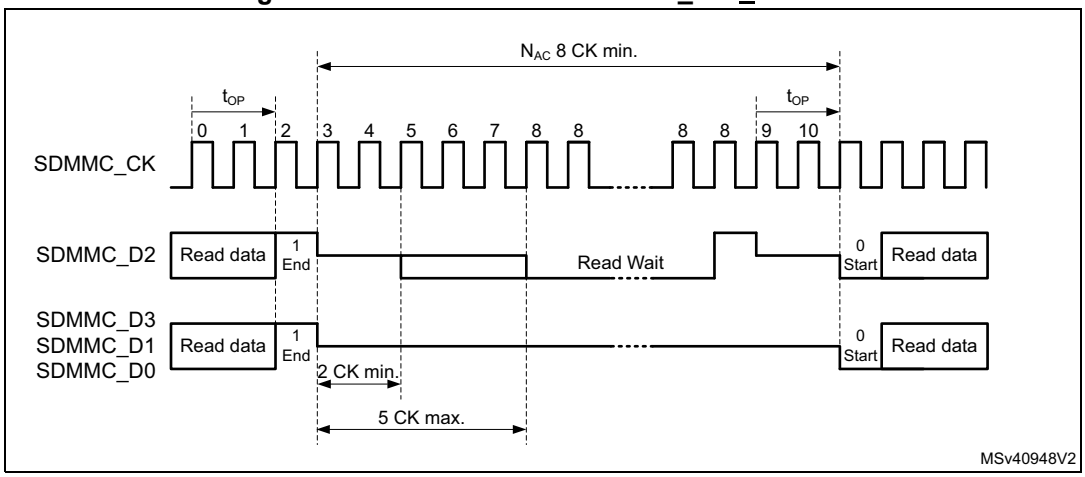
The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see *Figure 529*.

Figure 529. Read Wait with SDMMC_CK <_50 MHz



For SDR50 with a SDMMC_CK more than 50MHz, and DDR50, the card treats the Read Wait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host must raise SDMMC_D2 on the SDMMC_CK clock (see *Figure 530*).

Figure 530. Read Wait with SDMMC_CK ≥ 50 MHz



In Read Wait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the end bit of the current received data block CRC. The Read Wait signaling on SDMMC_D2 is removed when writing 1 to the RWSTOP bit. The DPSM remains in R_W state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a start bit from the card.

During the Read Wait signaling on SDMMC_D2 commands can be issued to the card. During the Read Wait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

48.5.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the end bit of the Stop Transmission command.

Table 390. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC stream write	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block write	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block write with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC stream read	The data transfer is stopped/aborted by sending the Stop Transmission command.

Table 390. CMD12 use cases (continued)

Data operation	Stop Transmission command CMD12 Description
SDMMC open ended multiple block read	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block read with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

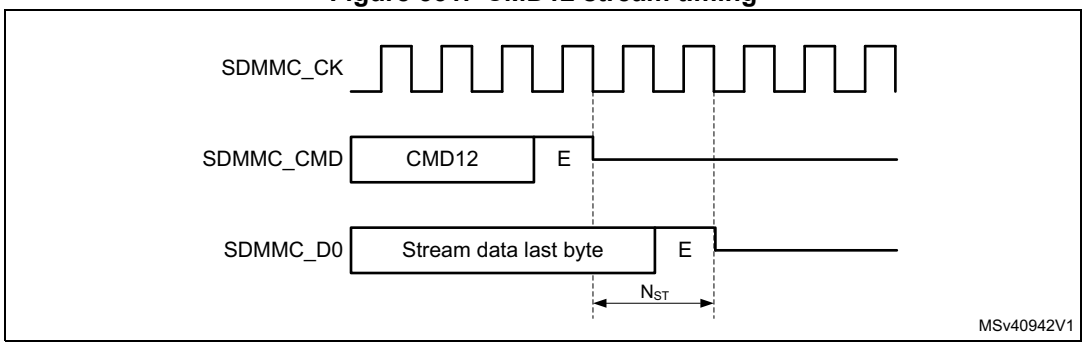
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   a) This causes the CPSM Abort signal to be generated when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   a) The card, when sending data, stops data transfer 2 cycles after the Stop Transmission command end bit.
   The card when no data is being sent, does not start sending any new data.
   b) The host, when sending data, sends one last data bit followed by an end bit after the Stop Transmission command end bit.
   The host when not sending data, does not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   a) When writing data to the card. On the CMDREND flag, firmware must stop writing data to the FIFO. Subsequently the FIFO must be reset with FIFORST, this flushes the FIFO.
   b) When reading data from the card. On the CMDREND flag, firmware must read the remaining data from the FIFO. Subsequently the FIFO must be reset with FIFORST.
4. When IDMAEN = 1, hardware takes care of the FIFO.
   a) When writing data to the card. On the CPSM Abort signal, hardware stops the IDMA and subsequently the FIFO is flushed.
   b) When reading data from the card. On the CPSM Abort signal, hardware instructs the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag is generated.

Stream operation and CMD12

To stop the stream transfer after the last byte to be transfered, the CMD12 end bit timing must be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data end bit and command end bit are aligned.
   a) When DATALENGTH > 5 bytes, Command CMD12 is waited in the CPSM to be aligned with the data transfer end bit.
   b) When DATALENGHT < 5 bytes, Command CMD12 is started before and the DPSM remains in the Wait_S state to align the data transfer end with the CMD12 end bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This causes the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 531. CMD12 stream timing



To stop the read stream transfer after the last byte, the CMD12 end bit timing must occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM (DATAEND flag).
   a) The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   a) CMD12 stops the card sending data.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 end bit must be sent after the last block end bit.

When writing data to the card the CMD12 end bit must be sent after the write data block CRC token end bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   a) The DPSM does not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   a) CMD12 sets the card to Idle mode.

When reading data from the card the CMD12 end bit must be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM (DATAEND flag).
   a) The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   a) CMD12 stops the Card sending more data and set the card to Idle mode. Any ongoing block transfer is aborted by the Card.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

48.5.3 Sleep (CMD5)

The *e*•MMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state
4. Vcc power supply is allowed to be switched off

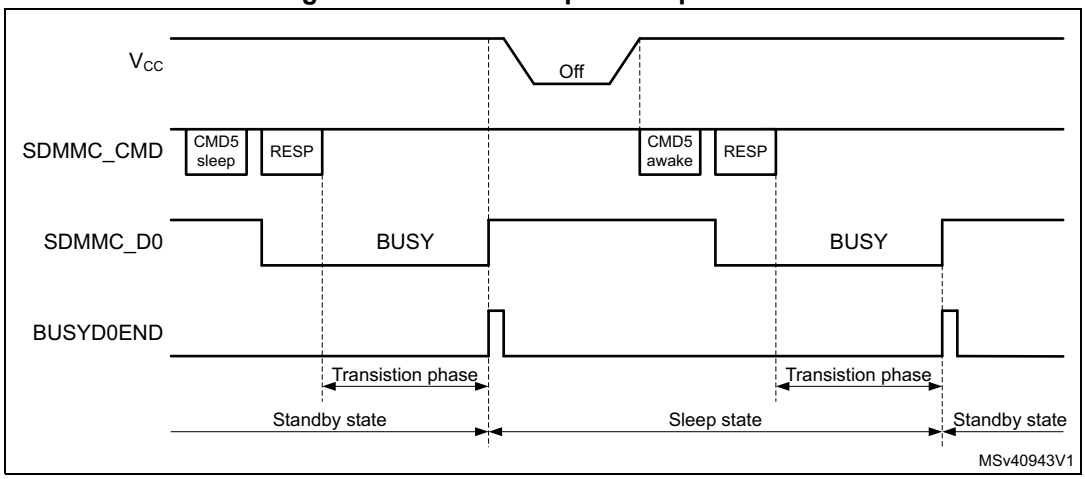
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply is allowed to be switched off only after the Sleep state has been reached. The Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.

Figure 532. CMD5 Sleep Awake procedure



48.5.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency must be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV must be set >1, and the SETCLKRX must select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):

- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a start bit. Upon detection of a start bit the card aborts the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):

- is waiting for a card interrupt service request response (start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one successfully sends its full interrupt service request response. If the host sends simultaneously, it loses sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV must be set >1, and SETCLKRX must select the sdmmc_io_in_ck.
2. Load CMD40 (GO_IRQ_STATE) in the command registers.
3. Enable wait for interrupt by setting WAITINT register bit.
4. Configure the CPSM to send a command immediately.
   a) This causes the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for a interrupt service request response.
5. To exit the wait for interrupt state (CPSM Wait state):
   a) Upon the detection of an interrupt service request response start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   b) To abort the interrupt mode the host clears the WAITINT register bit, which causes the host to send an interrupt service request response by itself. This moves the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response start bit collision the host loses the bus access after the transmission bit.

48.5.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the 2 boot operation functions:

1. Normal boot. (keeping CMD line low)
2. Alternative boot (sending CMD0 with argument 0xFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:

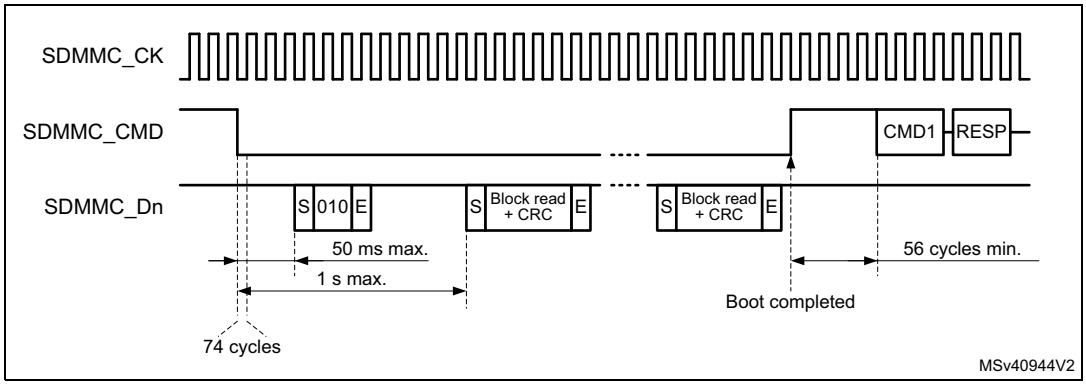
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.

Figure 533. Normal boot mode operation



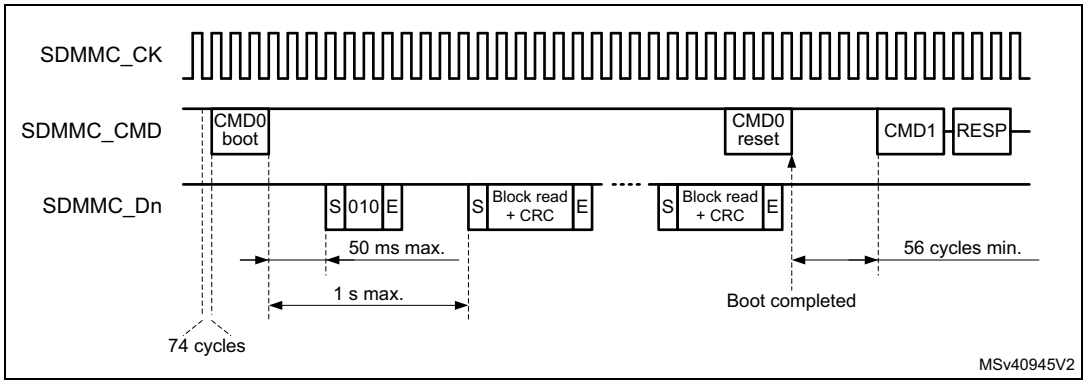
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN.This causes:
   – the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   – the ACK timeout to start.
   – DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   – when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   – when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. when all boot data has been received the DATAEND flag occurs.
   – when data CRC fails the DCRCFAIL flag is also generated.
   – when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty (RXFIFOE = 1) after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure is terminated by firmware clearing BOOTEN, which causes the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   a) If the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops data reception and disables the DPSM which triggers an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.

Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

Figure 534. Alternative boot mode operation



To perform the alternative boot procedure the following steps needed:

1. Move the SDMMC to power-off state, and reset the card
2. Move the SDMMC to power-on state. This guarantees the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of

boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:

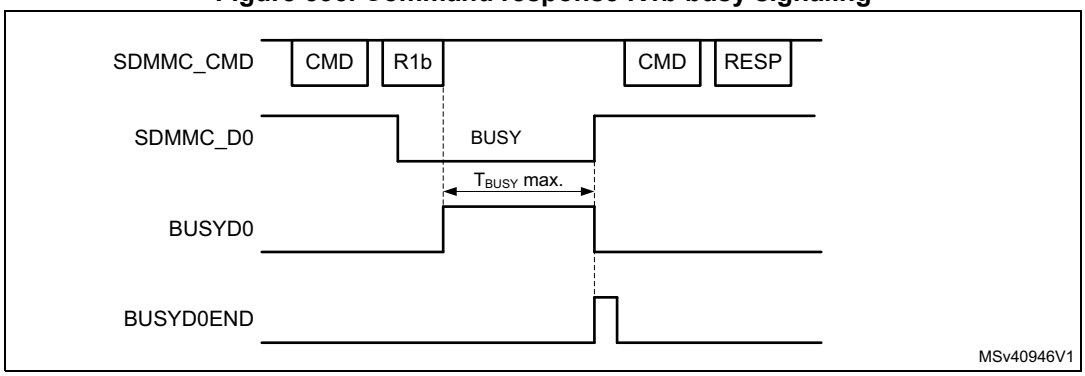
- the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
- the ACK timeout to start.
- DPSM to be enabled.

6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit must be cleared.
7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
8. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.
9. When last data has been received, read data from the FIFO until FIFO is empty (RXFIFOE = 1) after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.
10. The BOOTEN bit must be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This causes the CMDSENT flag to occur 56 cycles after the Command.
    - if the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops the data transfer and disable the DPSM which triggers an DABORT flag when enabled.
11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

48.5.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 535. Command response R1b busy signaling



The expected maximum busy time must be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag is set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  - If BUSYD0 signals not busy, signal busy release to the Firmware
  - If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to the firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

48.5.7 Reset and card cycle power

Reset

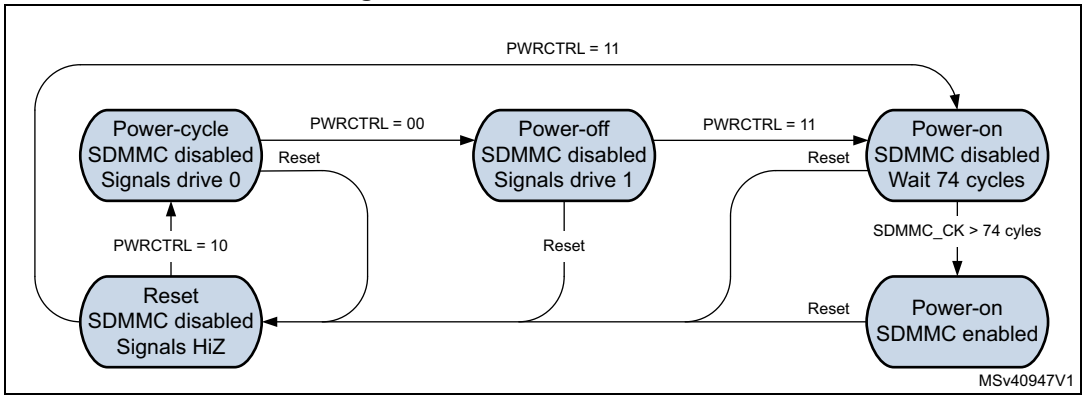
Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC must be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles are clocked after which the SDMMC is enabled and command and data can be transfered.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according *Figure 536.*.

Figure 536. SDMMC state control

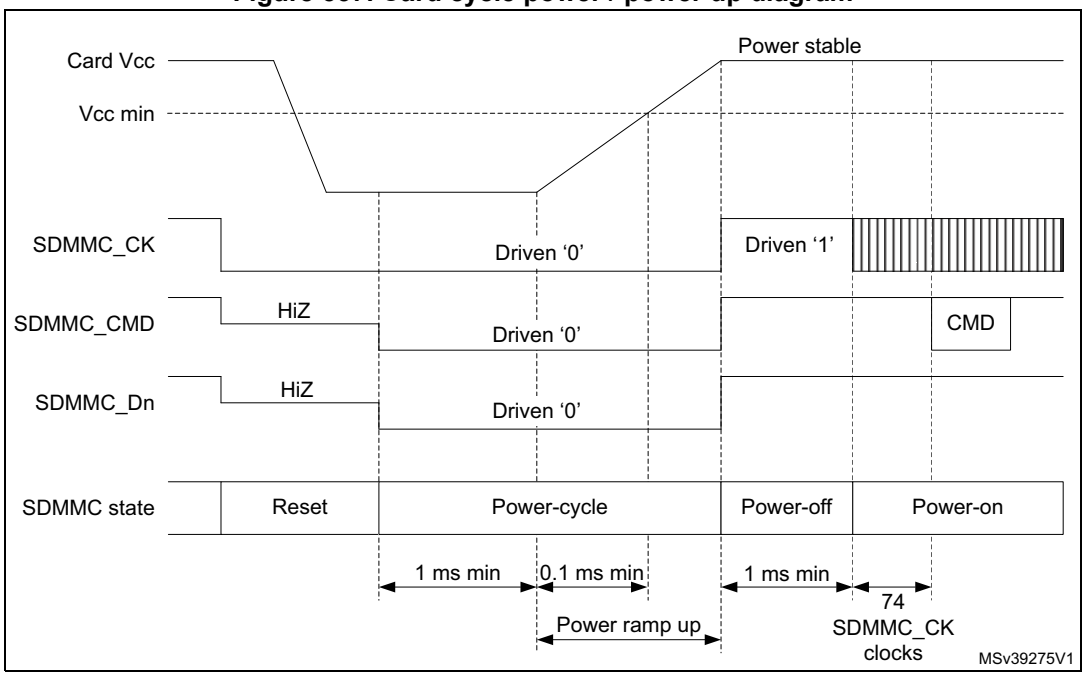


Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This resets the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This makes that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1 ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1 ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive “1”.
6. After the 1 ms delay set the SDMMC to power-on state in which the SDMMC_CK clock is enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.

Figure 537. Card cycle power / power up diagram



48.6 Hardware flow control

The hardware flow control during data transfer functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

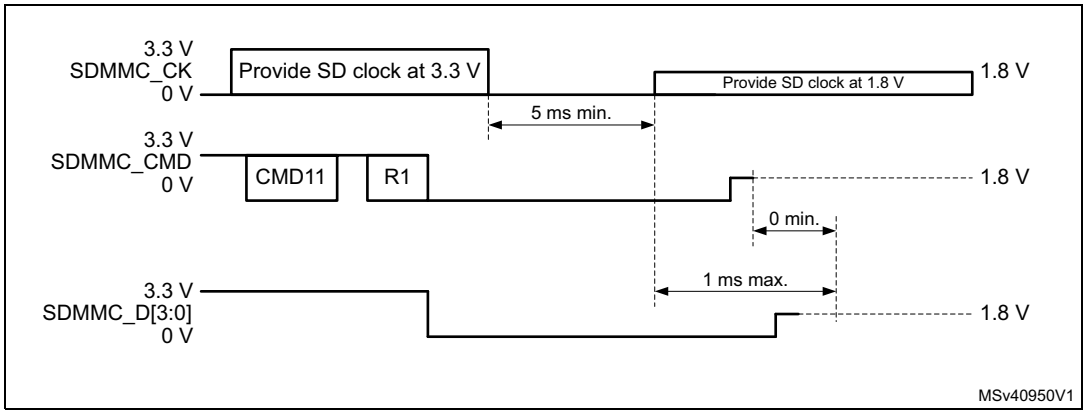
The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGHT has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

48.7 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, and DDR50) requires the support for 1.8V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

Figure 538. CMD11 signal voltage switch sequence



To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency must be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
   - if the response CRC is pass, the Voltage Switch procedure continues the host does no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK is stopped and the CKSTOP flag is set.
   - if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.
4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   - When the BUSYD0 is detected low the host firmware switches the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware continues to stop the SDMMC_CK by holding it low for at least 5 ms.
   - When the BUSYD0 is detected high the host aborts the Voltage Switch sequence and cycle power the card.
6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.
7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.
8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].

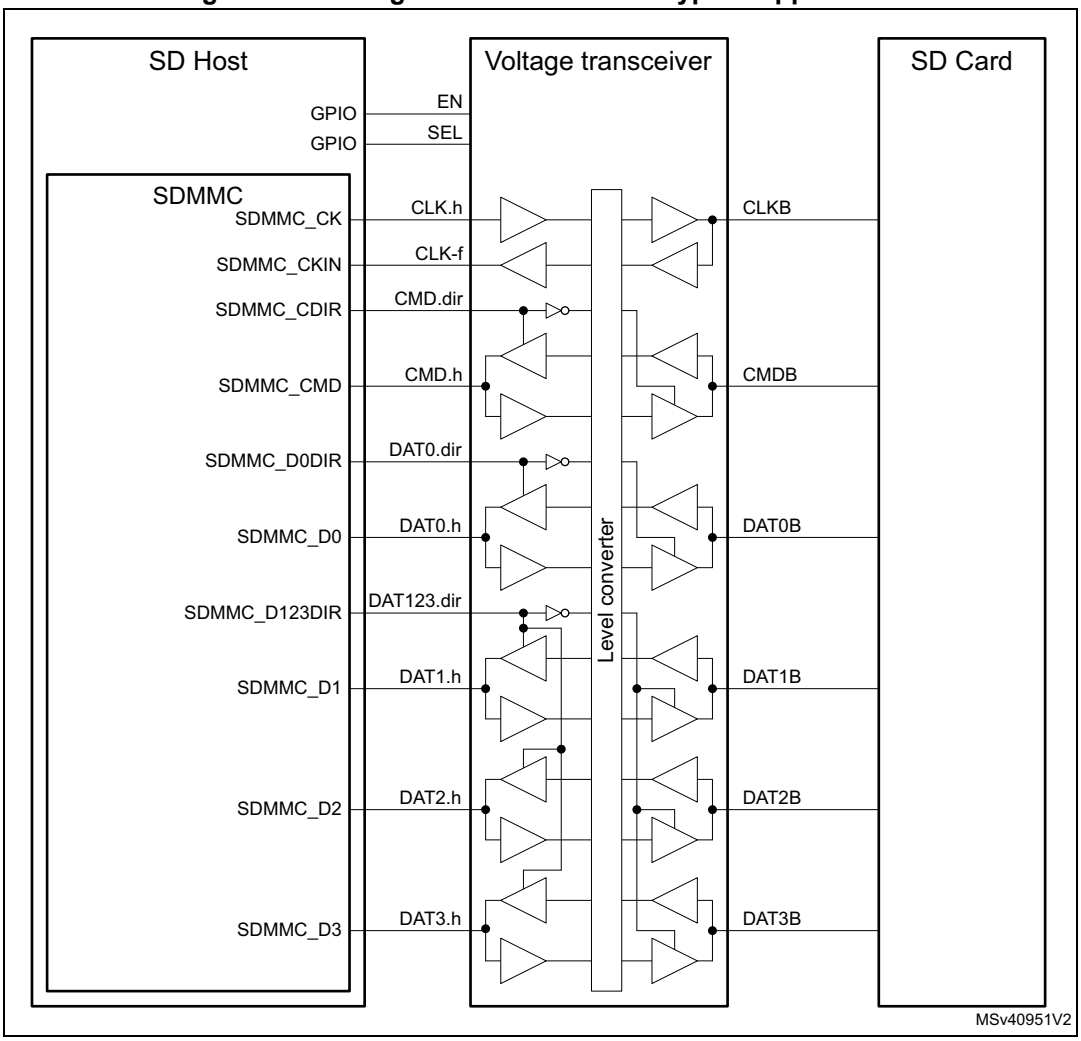
9. The host SDMMC hardware, 1 ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.
10. The host, on the VSWEND flag, checks SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
    – When BUSYD0 is detected high, Voltage Switch has been completed successfully.
    – When BUSYD0 is detected low, Voltage Switch has failed, the host cycles the card power.

The minimum 5 ms time to stop the SDMMC_CK is derived from the internal un-gated SDMMC_CK clock, which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The >5 ms time is counted by 2^{12} cycles (10.24 ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped is longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC checks the lines after >1 ms time which is counted by 2^9 cycles (1.28 ms @ 25 MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines is longer.

The signal voltage level is supported through an external voltage translation transceiver like STMicroelectronics ST6G3244ME.

Figure 539. Voltage switch transceiver typical application



To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

SDMMC_CKIN feedback input clock

SDMMC_CDIR I/O direction control for the CMD signal.

SDMMC_D0DIR I/O direction control for the SDMMC_D0 signal.

SDMMC_D123DIR I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals.

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.

48.8 SDMMC interrupts

Table 391. SDMMC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
SDMMC	Command response CRC fail	CCRCFAIL	CCRCFAILIE	CCRCFAILC	Yes
SDMMC	Data block CRC fail	DCRCFAIL	DCRCFAILIE	DCRCFAILC	Yes
SDMMC	Command response timeout	CTIMEOUT	CTIMEOUTIE	CTIMEOUTC	Yes
SDMMC	Data timeout	DTIMEOUT	DTIMEOUTIE	DTIMEOUTC	Yes
SDMMC	Transmit FIFO underrun	TXUNDERR	TXUNDERRIE	TXUNDERRC	Yes
SDMMC	Receive FIFO overrun	RXOVERR	RXOVERRIE	RXOVERRC	Yes
SDMMC	Command reponse received	CMDREND	CMDRENDIE	CMDRENDC	Yes
SDMMC	Comand sent	CMDSENT	CMDSENTIE	CMDSENTC	Yes
SDMMC	Data transfer ended	DATAEND	DATAENDIE	DATAENDC	Yes
SDMMC	Data transfer hold	DHOLD	DHOLDIE	DHOLDC	Yes
SDMMC	Data block sent or received	DBCKEND	DBCKENDIE	DBCKENDC	Yes
SDMMC	Data transfer aborted	DABORT	DABORTIE	DABORTC	Yes
SDMMC	Transmit FIFO half empty	TXFIFOHE	TXFIFOHEIE	n.a.	Yes
SDMMC	Receive FIFO half full	RXFIFOHF	RXFIFOHFIE	n.a.	Yes
SDMMC	Transmit FIFO full	TXFIFOF	n.a.	n.a.	Yes
SDMMC	Receive FIFO full	RXFIFOF	RXFIFOFIE	n.a.	Yes
SDMMC	Transmit FIFO empty	TXFIFOE	TXFIFOEIE	n.a.	Yes
SDMMC	Receive FIFO empty	RXFIFOE	n.a.	n.a.	Yes

Table 391. SDMMC interrupts (continued)

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
SDMMC	Command response end of busy	BUSYD0END	BUSYD0ENDIE	BUSYD0ENDC	Yes
SDMMC	SDIO interrupt	SDIOIT	SDIOITIE	SDIOITC	Yes
SDMMC	Boot acknowledgment fail	ACKFAIL	ACKFAILIE	ACKFAILC	Yes
SDMMC	Boot acknowledgment timeout	ACKTIMEOUT	ACKTIMEOUTIE	ACKTIMEOUTC	Yes
SDMMC	Voltage switch timing	VSWEND	VSWENDIE	VSWENDC	Yes
SDMMC	SDMM_CK stopped in voltage switch	CKSTOP	CKSTOPIE	CKSTOPC	Yes
SDMMC	IDMA transfer error	IDMATE	IDMATEIE	IDMATEC	Yes
SDMMC	IDMA buffer transfer complete	IDMABTC	IDMABTCIE	IDMABTCC	Yes

48.9 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and halfword (16-bit) accesses trigger an AHB bus error.

48.9.1 SDMMC power control register (SDMMC_POWER)

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.	Res.	DIR POL	VSWI TCHEN	VSWI TCH	PWRCTRL[1:0]	
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).

0: Voltage transceiver IOs driven as output when direction signal is low.

1: Voltage transceiver IOs driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

This bit is used to stop the SDMMC_CK after the voltage switch command response:

0: SDMMC_CK clock kept unchanged after successfully received command response.

1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:

0: Voltage switch sequence not started and not active.

1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).

These bits are used to define the functional state of the SDMMC signals:

00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.

01: Reserved. (When written 01, PWRCTRL value does not change)

10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.

11: Power-on: the card is clocked, The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation.

Any further write is ignored, PWRCTRL value keeps 11.

48.9.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004

Reset value: 0x0000 0000

The SDMMC_CLKCR register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUS SPEED	DDR	HWFC_EN	NEG EDGE
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WID BUS[1:0]		Res.	PWR SAV	Res.	Res.	CLKDIV[9:0]									
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection

These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: sdmmc_io_in_ck selected as receive clock

01: SDMMC_CKIN feedback clock selected as receive clock

10: Reserved

11: Reserved (select sdmmc_io_in_ck)

Bit 19 **BUSSPEED**: Bus speed for selection of SDMMC operating modes

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected

1: SDR50, DDR50 bus speed mode selected.

Bit 18 **DDR**: Data rate signaling selection

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

DDR rate must only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).

DDR rate must only be selected with clock division >1. (CLKDIV > 0)

0: SDR Single data rate signaling

1: DDR double data rate signaling

Bit 17 **HWFC_EN**: Hardware flow control enable

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: Hardware flow control is disabled

1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, please see SDMMC status register definition in *Section 48.9.11*.

Bit 16 **NEGEDGE**: SDMMC_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC_CK falling edge.

0: When clock division >1 (CLKDIV > 0) & DDR = 0:

- Command and data changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

When clock division >1 (CLKDIV > 0) & DDR = 1:

- Command changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- Data changed on the sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: When clock division >1 (CLKDIV > 0) & DDR = 0:

- Command and data changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) & DDR = 1:

- Command changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.
- Data changed on the SDMMC_CK falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC_D0 used (Does not support DDR)

01: 4-bit wide bus mode: SDMMC_D[3:0] used

10: 8-bit wide bus mode: SDMMC_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled

1: SDMMC_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc_ker_ck) and the output clock (SDMMC_CK): SDMMC_CK frequency = sdmmc_ker_ck / [2 * CLKDIV].

0x000: SDMMC_CK frequency = sdmmc_ker_ck / 1 (Does not support DDR)

0x001: SDMMC_CK frequency = sdmmc_ker_ck / 2

0x002: SDMMC_CK frequency = sdmmc_ker_ck / 4

0x0XX: etc..

0x080: SDMMC_CK frequency = sdmmc_ker_ck / 256

0xXXX: etc..

0x3FF: SDMMC_CK frequency = sdmmc_ker_ck / 2046

Note: 1 While the SD/SDIO card or e•MMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

2 The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

3 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the Read Wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

48.9.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

The SDMMC_ARGR register contains a 32-bit command argument, which is sent to a card as part of a command message.

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

Bits 31:0 **CMDARG[31:0]**: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

48.9.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

The SDMMC_CMDR register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD SUS PEND
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT EN	BOOT MODE	DT HOLD	CPSM EN	WAITP END	WAIT INT	WAITRESP[1:0]		CMD STOP	CMD TRANS	CMDINDEX[5:0]					
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 **CMDSUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
CMDSUSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS=0.
CMDSUSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF=1.

Bit 15 **BOOTEN**: Enable boot mode procedure

0: Boot mode procedure disabled
1: Boot mode procedure enabled

Bit 14 **BOOTMODE**: Select the boot mode procedure to be used

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
0: Normal boot mode procedure selected
1: Alternative boot mode procedure selected.

Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM

If this bit is set, the DPSM does not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit

This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
If this bit is set, the CPSM is enabled.
When DTEN = 1, no command is transfered nor boot procedure is started. CPSMEN is cleared to 0.
During Read Wait with SDMMC_CK stopped no command is sent and CPSMEN is kept 0.

Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM

This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
WAITPEND is only taken into account when DTMODE = *e*•MMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10 **WAITINT**: CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
If this bit is cleared in the CPSM Wait state, it causes the abort of the interrupt mode.

Bits 9:8 **WAITRESP[1:0]**: Wait for response bits

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
00: No response, expect CMDSENT flag
01: Short response, expect CMDREND or CCRCFAIL flag
10: Short response, expect CMDREND flag (No CRC)
11: Long response, expect CMDREND or CCRCFAIL flag

Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals abort to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

Note: 1 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2 MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses,136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

48.9.5 SDMMC command response register (SDMMC_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

The SDMMC_RESPCMDR register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	RESPCMD[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bit field. Contains the command index of the last command response received.

48.9.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)

Reset value: 0x0000 0000

The SDMMC_RESP1/2/3/4R registers contain the status of a card, which is part of the received response.

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

Bits 31:0 **CARDSTATUSx[31:0]**: see *Table 392*

The card status size is 32 or 128 bits, depending on the response type.

Table 392. Response type and SDMMC_RESPxR registers

Register(1)	Short response	Long response
SDMMC_RESP1R	Card status[31:0]	Card status [127:96]
SDMMC_RESP2R	all 0	Card status [95:64]
SDMMC_RESP3R	all 0	Card status [63:32]
SDMMC_RESP4R	all 0	Card status [31:0](2)

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

48.9.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

The SDMMC_DTIMER register contains the data timeout period, in card bus clock periods.

A counter loads the value from the SDMMC_DTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

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

Bits 31:0 **DATATIME[31:0]**: Data and R1b busy timeout period

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

Data and R1b busy timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

48.9.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

The SDMMC_DLENR register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:16]								
							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
DATALENGTH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATALENGTH[24:0]**: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transfered)

When DATALENGTH = 0 no data are transfered, when requested by a CPSMEN and CMDTRANS = 1 also no command is transfered. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.

48.9.9 SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C

Reset value: 0x0000 0000

The SDMMC_DCTRL register control the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FIFO RST	BOOT ACK EN	SDIO EN	RW MOD	RW STOP	RW START	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FIFORST**: FIFO reset, flushes any remaining data

This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit only takes effect when a transfer error or transfer hold occurs.

0: FIFO not affected.

1: Flush any remaining data and reset the FIFO pointers. This bit is automatically cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12 **BOOTACKEN**: Enable the reception of the boot acknowledgment

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Boot acknowledgment disabled, not expected to be received

1: Boot acknowledgment enabled, expected to be received

Bit 11 **SDIOEN**: SD I/O interrupt enable functions

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10 **RWMOD**: Read Wait mode

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Read Wait control using SDMMC_D2

1: Read Wait control stopping SDMMC_CK

Bit 9 **RWSTOP**: Read Wait stop

This bit is written by firmware and auto cleared by hardware when the DPSM moves from the R_W state to the Wait_R or Idle state.

0: No Read Wait stop.

1: Enable for Read Wait stop when DPSM is in the R_W state.

Bit 8 **RWSTART**: Read Wait start

If this bit is set, Read Wait operation starts.

Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
Define the data block length when the block data transfer mode is selected:
0000: (0 decimal) lock length = 2^0 = 1 byte
0001: (1 decimal) lock length = 2^1 = 2 bytes
0010: (2 decimal) lock length = 2^2 = 4 bytes
0011: (3 decimal) lock length = 2^3 = 8 bytes
0100: (4 decimal) lock length = 2^4 = 16 bytes
0101: (5 decimal) lock length = 2^5 = 32 bytes
0110: (6 decimal) lock length = 2^6 = 64 bytes
0111: (7 decimal) lock length = 2^7 = 128 bytes
1000: (8 decimal) lock length = 2^8 = 256 bytes
1001: (9 decimal) lock length = 2^9 = 512 bytes
1010: (10 decimal) lock length = 2^{10} = 1024 bytes
1011: (11 decimal) lock length = 2^{11} = 2048 bytes
1100: (12 decimal) lock length = 2^{12} = 4096 bytes
1101: (13 decimal) lock length = 2^{13} = 8192 bytes
1110: (14 decimal) lock length = 2^{14} = 16384 bytes
1111: (15 decimal) reserved
When DATALENGTH is not a multiple of DBLOCKSIZE, the transfered data is truncated at a multiple of DBLOCKSIZE. (None of the remaining data are transfered.)
When DDR = 1, DBLOCKSIZE = 0000 must not be used. (No data are transfered)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
00: Block data transfer ending on block count.
01: SDIO multibyte data transfer.
10: *e*•MMC Stream data transfer. (WIDBUS must select 1-bit wide bus mode)
11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: From host to card.
1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by Hardware when data transfer completes.
This bit must only be used to transfer data when no associated data transfer command is used, i.e. must not be used with SD or *e*•MMC cards.
0: Do not start data transfer without CPSM data transfer command.
1: Start data transfer without CPSM data transfer command.

48.9.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

The SDMMC_DCNTR register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then

moves to the Idle state and when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:16]								
							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
DATACOUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register should be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transfered.

48.9.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

The SDMMC_STAR register is a read-only register. It contains two types of flag:

- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTC	IDMA TE	CK STOP	VSW END	ACK TIME OUT	ACK FAIL	SDIOIT	BUSY D0END	BUSY D0	RX FIFOE	TX FIFOE	RX FIFOF	TX FIFOF
			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
RX FIFO HF	TX FIFO HE	CPSM ACT	DPSM ACT	DA BORT	DBCK END	DHOLD	DATA END	CMD SENT	CMDR END	RX OVERR	TX UNDER R	D TIME OUT	C TIME OUT	DCRC FAIL	CCRC FAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 27 **IDMATE**: IDMA transfer error

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 26 **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 25 **VSWEND**: Voltage switch critical timing section completion
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected
This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
0: card SDMMC_D0 signal does NOT signal change from busy to not busy.
1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response
This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.
0: card signals not busy on SDMMC_D0.
1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty
This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full
This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full
There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty
At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active, i.e. not in Idle state
This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active, i.e. not in Idle state
This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 10 **DBCKEND**: Data block sent/received
DBCKEND is set when:
- CRC check passed and DPSM moves to the R_W state
or
- IDMAEN = 0 and transmit data transfer hold and DATACOUNT >0 and DPSM moves to Wait_S.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHOLD**: Data transfer Hold
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly
DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit data transfer hold.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENT**: Command sent (no response required)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Note: FIFO interrupt flags must be masked in SDMMC_MASKR when using IDMA mode.

48.9.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

The SDMMC_ICR register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCC	IDMA TEC	CK STOPC	VSW ENDC	ACK TIME OUTC	ACK FAILC	SDIO ITC	BUSY D0 ENDC	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.	D ABORT C	DBCK ENDC	DHOLD C	DATA ENDC	CMD SENTC	CMDR ENDC	RX OVERR C	TX UNDER RC	D TIME OUTC	C TIME OUTC	DCRC FAILC	CCRC FAILC
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCC**: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 **IDMATEC**: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 **CKSTOPC**: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 **VSWENDC**: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 **ACKTIMEOUTC**: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 **ACKFAILC**: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
Set by software to clear the BUSYD0END flag.
0: BUSYD0END not cleared
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
Set by software to clear the DABORT flag.
0: DABORT not cleared
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
Set by software to clear the DHOLD flag.
0: DHOLD not cleared
1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTC**: CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREND flag clear bit
Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit
Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared

Bit 2 **CTIMEOUTC**: CTIMEOUT flag clear bit
Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared

Bit 1 **DCRCFAILC**: DCRCFAIL flag clear bit
Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared

Bit 0 **CCRCFAILC**: CCRCFAIL flag clear bit
Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

48.9.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

The interrupt mask register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCIE	Res.	CK STOP IE	VSW ENDIE	ACK TIME OUTIE	ACK FAILIE	SDIO ITIE	BUSY D0 ENDIE	Res.	Res.	TX FIFO EIE	RX FIFO FIE	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
RX FIFO HFIE	TX FIFO HEIE	Res.	Res.	DA BORT IE	DBCK ENDIE	DHOLD IE	DATA ENDIE	CMD SENT IE	CMDR ENDIE	RX OVER RIE	TX UNDER RIE	D TIME OUTIE	C TIME OUTIE	DCRC FAILIE	CCRC FAILIE
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable
Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
0: IDMA buffer transfer complete interrupt disabled
1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage Switch clock stopped interrupt enable
Set and cleared by software to enable/disable interrupt caused by Voltage Switch clock stopped.
0: Voltage Switch clock stopped interrupt disabled
1: Voltage Switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
0: Voltage switch critical timing section completion interrupt disabled
1: Voltage switch critical timing section completion interrupt enabled

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment Fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment Fail.
0: Acknowledgment Fail interrupt disabled
1: Acknowledgment Fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO Mode interrupt received interrupt disabled
1: SDIO Mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

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

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled

Bit 9 **DHOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **DATAENDIE**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

48.9.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040

Reset value: 0x0000 0000

The SDMMC_ACKTIMER register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from the SDMMC_ACKTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:16]								
							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
ACKTIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
Boot acknowledgment timeout period expressed in card bus clock periods.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

48.9.15 SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x =0 to 15)

Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

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

Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data

This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).

The FIFO data occupies 16 entries of 32-bit words.

48.9.16 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO.

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

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IDMABACT**: Double buffer mode active buffer indication

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). When IDMA is enabled this bit is toggled by hardware.

0: When IDMA is enabled, uses buffer0 and firmware write access to IDMABASE0 is prohibited.

1: When IDMA is enabled, uses buffer1 and firmware write access to IDMABASE1 is prohibited.

Bit 1 **IDMABMODE**: Buffer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Single buffer mode.

1: Double buffer mode.

Bit 0 **IDMAEN**: IDMA enable

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: IDMA disabled

1: IDMA enabled

48.9.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

The SDMMC_IDMABSIZER register contains the buffers size when in double buffer configuration.

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

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:5 **IDMABNDT[7:0]**: Number of bytes per buffer

This 8-bit value must be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.

Example: IDMABNDT = 0x01: buffer size = 8 words = 32 bytes.

Example: IDMABNDT = 0x80: buffer size = 1024 words = 4 Kbyte

These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 4:0 Reserved, must be kept at reset value.

48.9.18 SDMMC IDMA buffer 0 base address register (SDMMC_IDMABASE0R)

Address offset: 0x058

Reset value: 0x0000 0000

The SDMMC_IDMABASE0R register contains the memory buffer base address in single buffer configuration and the buffer 0 base address in double buffer configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABASE0[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
IDMABASE0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **IDMABASE0[31:0]**: Buffer 0 memory base address bits [31:2], must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1) and memory buffer 0 is inactive (IDMABACT = ‘1’).

48.9.19 SDMMC IDMA buffer 1 base address register (SDMMC_IDMABASE1R)

Address offset: 0x05C

Reset value: 0x0000 0000

The SDMMC_IDMABASE1R register contains the double buffer configuration second buffer memory base address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABASE1[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
IDMABASE1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **IDMABASE1[31:0]**: Buffer 1 memory base address, must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1) and memory buffer 1 is inactive (IDMABACT = ‘0’).

48.9.20 SDMMC register map

The following table summarizes the SDMMC registers.

Table 393. SDMMC register 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	SDMMC_ POWER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH	PWRCTRL[1:0]	
	Reset value																												0	0	0	0	0
0x004	SDMMC_ CLKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUSSPEED	DDR	HWFC_EN	NEGEDGE	WIDBUS[1:0]		Res.	PWRSAV	Res.	Res.	CLKDIV[9:0]									
	Reset value											0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0
0x008	SDMMC_ ARGR	CMDARG[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	SDMMC_ CMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSUSPEND	BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDSTOP	CMDTRANS	CMDINDEX[5:0]					
	Reset value																																
0x010	SDMMC_ RESPCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
	Reset value																											0	0	0	0	0	0
0x014	SDMMC_ RESP1R	CARDSTATUS1[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	SDMMC_ RESP2R	CARDSTATUS2[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
0x01C	SDMMC_ RESP3R	CARDSTATUS3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SDMMC_ RESP4R	CARDSTATUS4[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	SDMMC_ DTIMER	DATATIME[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	SDMMC_ DLENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:0]																								
	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

Table 393. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	**SDMMC_ DCTRLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFORST	BOOTACKEN	SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCK SIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	**SDMMC_ DCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:0]																								
	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
0x034	**SDMMC_ STAR**	Res.	Res.	Res.	IDMABTC	IDMATE	CKSTOP	VSWEND	ACKTIMEOUT	ACKFAIL	SDIOIT	BUSYD0END	BUSYD0	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	CPSMACT	DPSMACT	DABORT	DBCKEND	DHOLD	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
	Reset value				0	0	0	0	0	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	**SDMMC_ ICR**	Res.	Res.	Res.	IDMABTCC	IDMATEC	CKSTOPC	VSWENDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
	Reset value				0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0
0x03C	**SDMMC_ MASKR**	Res.	Res.	Res.	IDMABTCIE	Res.	CKSTOPIE	VSWENDIE	ACKTIMEOUTIE	ACKFAILIE	SDIOITIE	BUSYD0ENDIE	Res.	Res.	TXFIFOEIE	RXFIFOFIE	Res.	RXFIFOHFIE	TXFIFOHEIE	Res.	Res.	DABORTIE	DBCKENDIE	DHOLDIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	Reset value				0		0	0	0	0	0	0			0	0		0	0			0	0	0	0	0	0	0	0	0	0	0	0
0x040	**SDMMC_ ACKTIMER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:0]																								
	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
0x044 - 0x04C	**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.
0x050	**SDMMC_ IDMACTRLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABACT	IDMABMODE	IDMAEN
	Reset value																														0	0	0
0x054	**SDMMC_ IDMABSIZER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[7:0]								Res.	Res.	Res.	Res.	Res.
	Reset value																				0	0	0	0	0	0	0	0					
0x058	**SDMMC_ IDMABASE0R**	IDMABASE0[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	**SDMMC_ IDMABASE1R**	IDMABASE1[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 - 0x07C	**Reserved**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x080 + 0x04 * x, (x=0..15)	**SDMMC_ FIFOR**	FIFODATA[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 2.3 on page 87* for the register boundary addresses.

49 FD controller area network (FDCAN)

49.1 Introduction

The controller area network (CAN) subsystem (see *Figure 540*) 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.

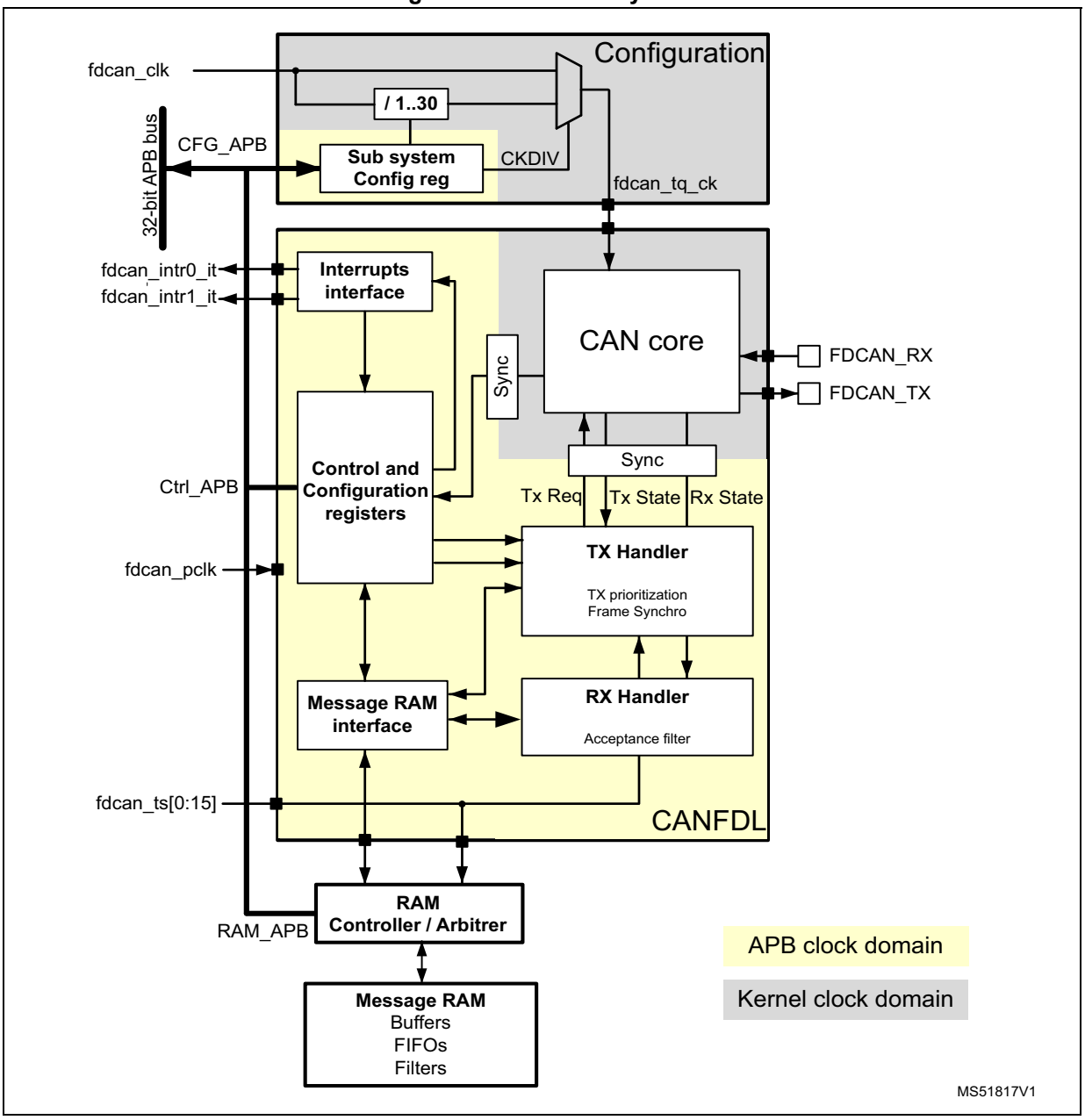
A 0.8-Kbyte message RAM per FDCAN instance implements filters, receive FIFOs, transmit event FIFOs and transmit FIFOs.

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 394* and *Figure 540*.

Table 394. 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 540. CAN subsystem

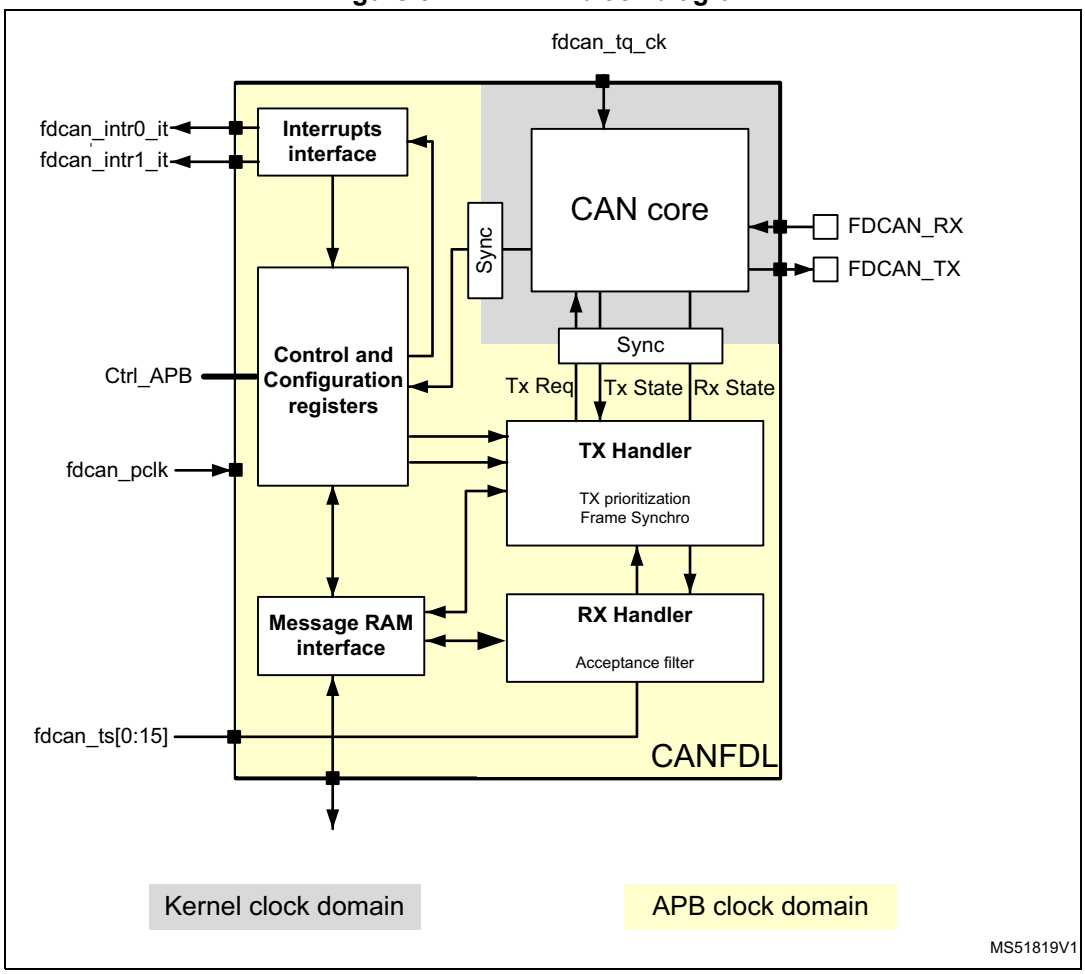


49.2 FDCAN main features

- 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

49.3 FDCAN functional description

Figure 541. 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.

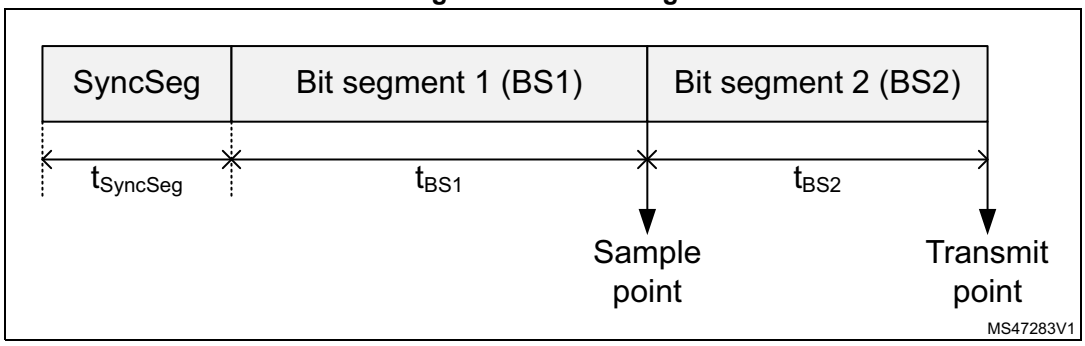
49.3.1 Bit timing

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 542*, 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 542. 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 542* 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.

49.3.2 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 395*.

Table 395. 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).

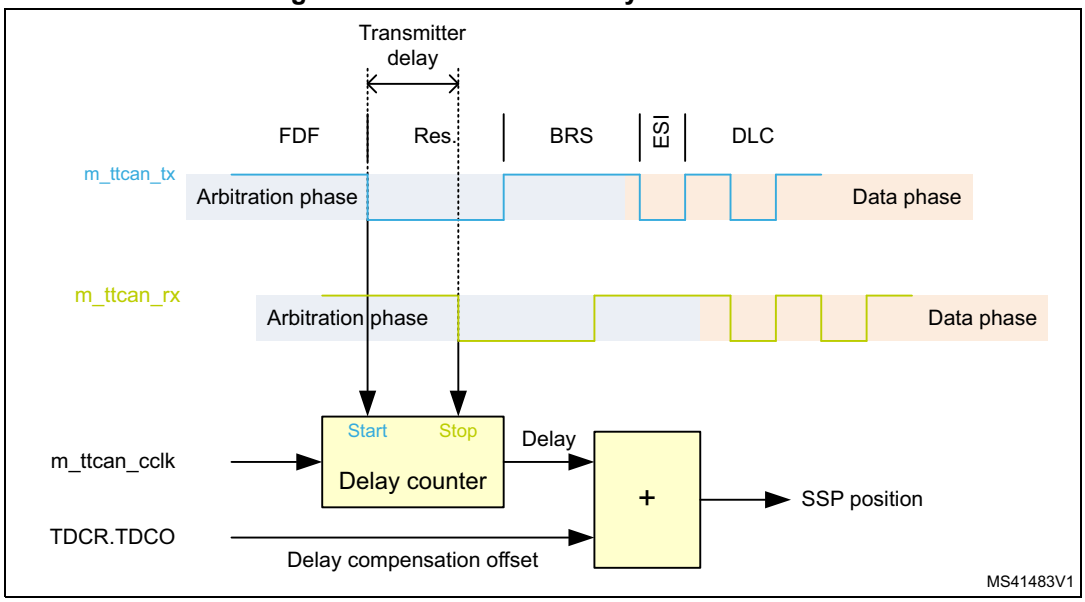
PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

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 543. 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.

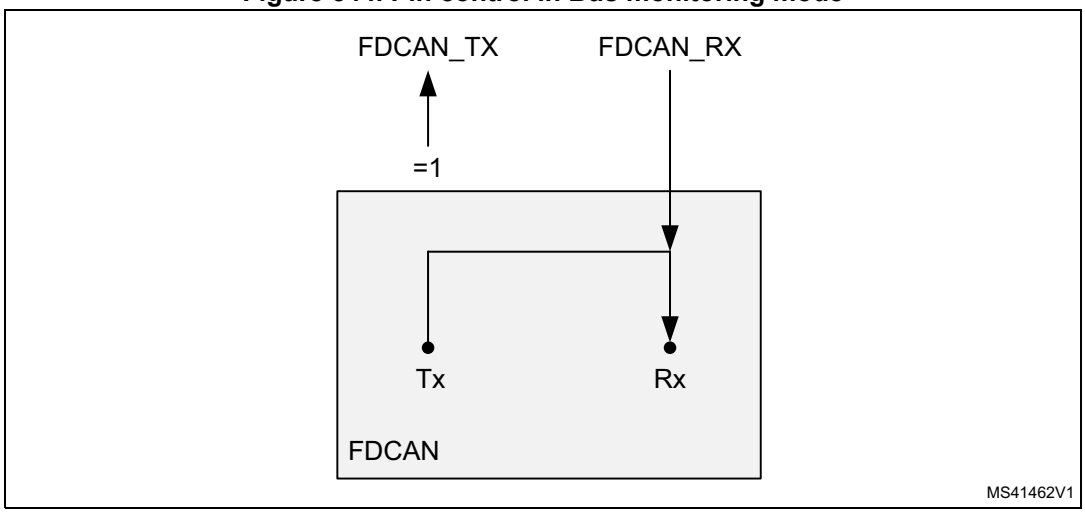
Note: The Restricted operation mode must not be combined with the Loop Back mode (internal or external).

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 544* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in Bus monitoring mode.

Figure 544. 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

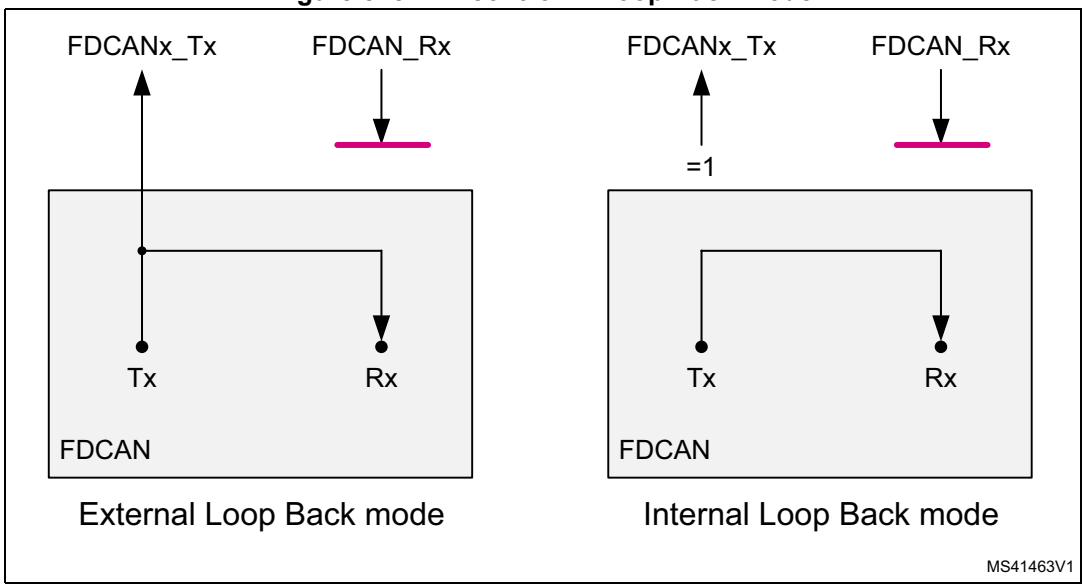
The FDCAN can be set in External Loop Back mode by programming TEST.LBCK to 1. In Loop Back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 545* shows the connection of transmit and receive signals FDCAN_TX 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 545* shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of Internal Loop Back mode.

Figure 545. 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.

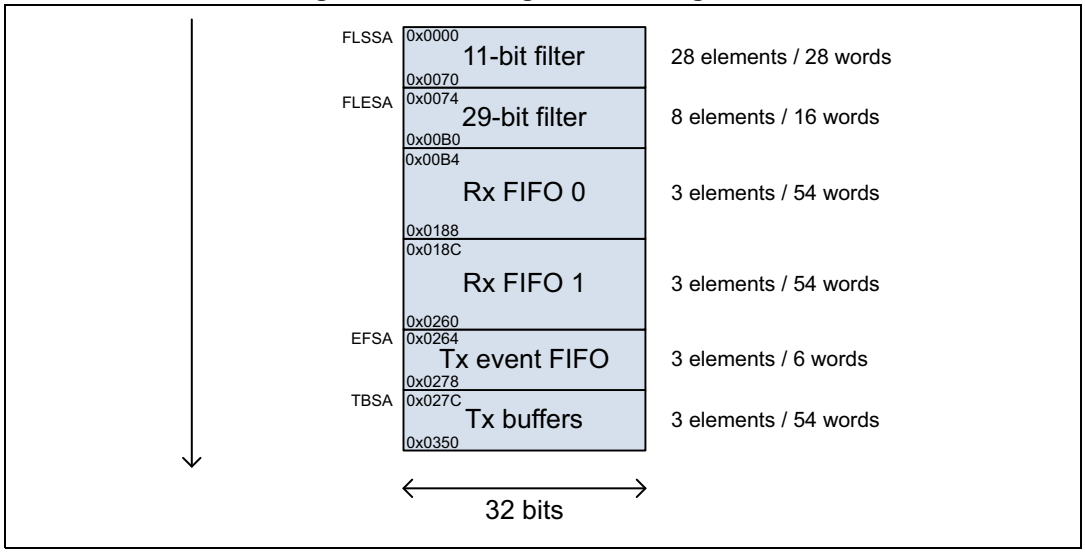
When the counter reaches 0, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

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.

49.3.3 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 546*.

Figure 546. 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 546*)
- 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.

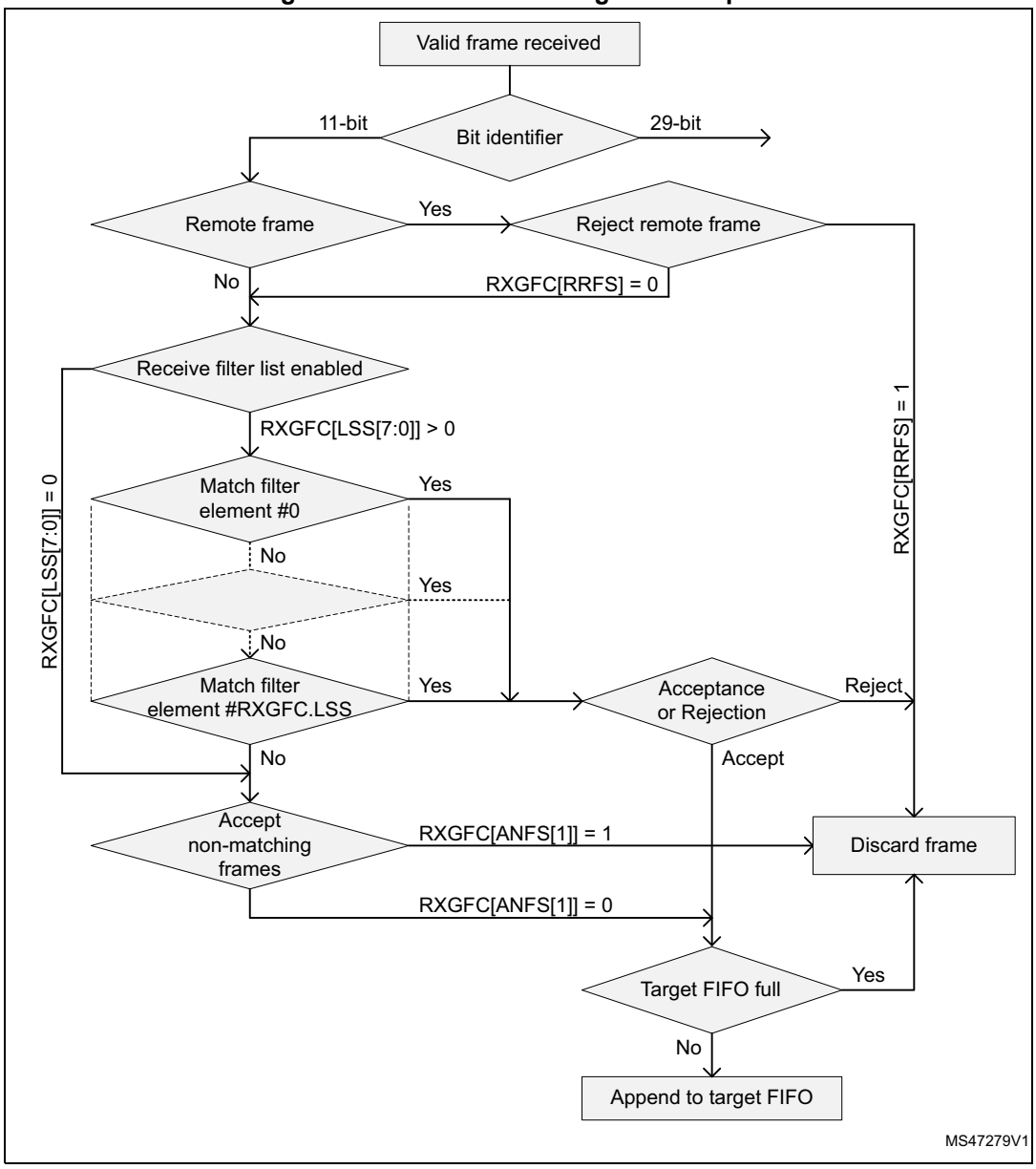
In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are 0, all Message IDs match.

Standard message ID filtering

Figure 547 shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID filter element is described in *Section 49.3.8*.

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 547. Standard Message ID filter path

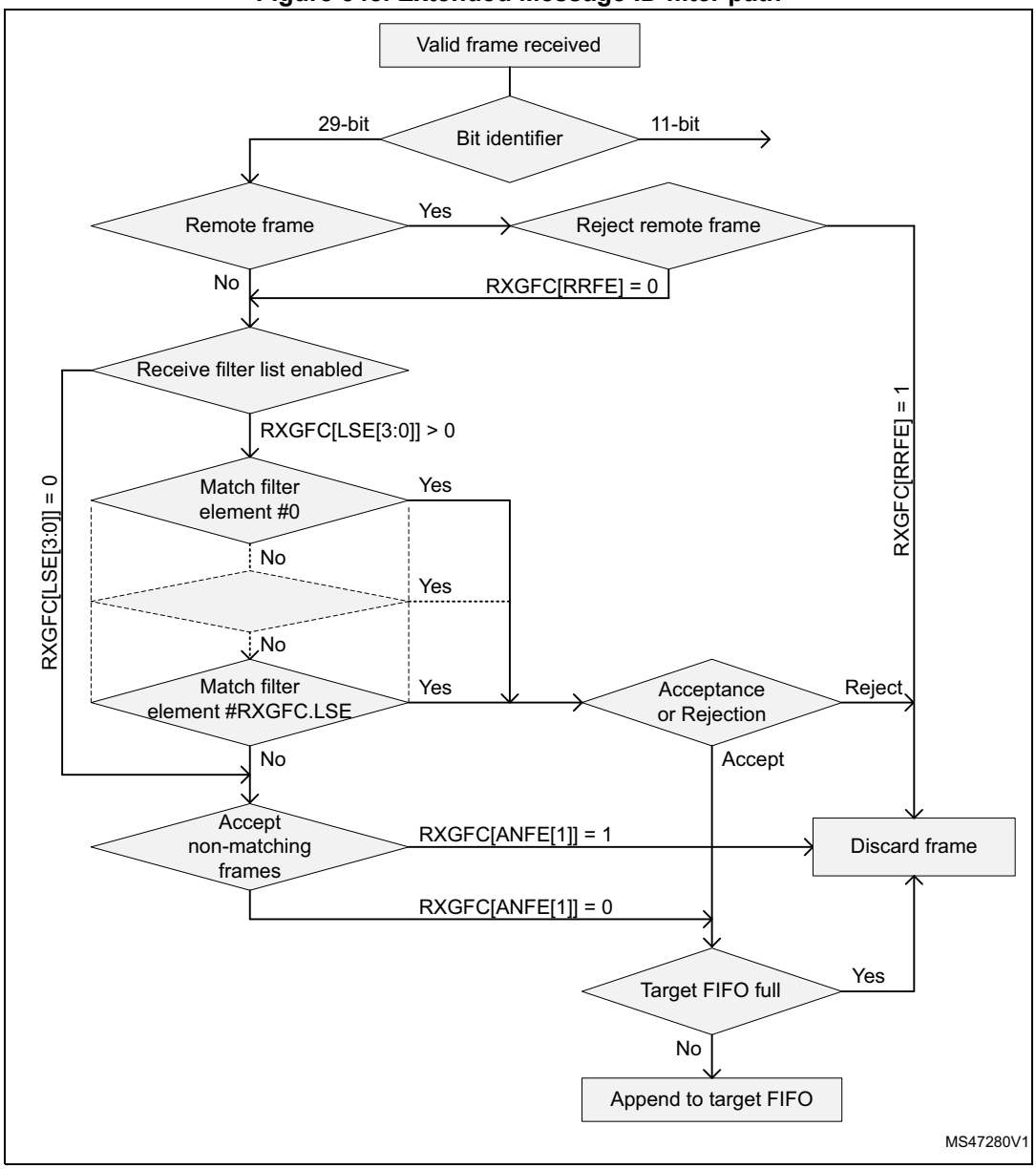


Extended message ID filtering

Figure 548 shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID filter element is described in *Section 49.3.9*.

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 548. 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 49.3.5*.

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 396. 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.

49.3.4 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 49.4.23* and *Section 49.4.25*. 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.

49.3.5 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 397*, the description is provided in *Table 398*.

Table 397. Rx FIFO element

Bit	31					24	23					16	15	8	7	0
R0	ESI	XTD	RTR	ID[28:0]												
R1	ANMF	FIDX[6:0]					Res.	FDF	BRS	DLC[3:0]			RXTS[15:0]			
R2	DB3[7:0]						DB2[7:0]						DB1[7:0]		D[7:0]	
R3	DB7[7:0]						DB6[7:0]						DB5[7:0]		DB4[7:0]	
⋮	⋮						⋮						⋮			
Rn	DBm[7:0]						DBm-1[7:0]						DBm-2[7:0]		DBm-3[7:0]	

The element size configured for storage of CAN FD messages is set to 64 bytes data field.

Table 398. 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 398. 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 398. 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

49.3.6 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 399. Tx buffer and FIFO element

Bit	31 24	23 16	15 8	7 0
T0	ESI | XTD | RTR | ID[28:0]			
T1	MM[7:0]	EFC | Res. | FDF | BPS | DLC[3:0]	Res.	
T2	DB3[7:0]	DB2[7:0]	DB1[7:0]	D[7:0]
T3	DB7[7:0]	DB6[7:0]	DB5[7:0]	DB4[7:0]
⋮	⋮	⋮	⋮	
Tn	DBm[7:0]	DBm-1[7:0]	DBm-2[7:0]	DBm-3[7:0]

Table 400. 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 400. 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.

49.3.7 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 401. Tx event FIFO element

Bit	31 24	23 16	15 8	7 0
E0	ESI | XTD | RTR	ID[28:0]		
E1	MM[7:0]	ET[1:0] | EDL | BRS | DLC[3:0]	TXTS[15:0]	

Table 402. 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 402. 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].

49.3.8 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 403. Standard Message ID Filter element

Bit	31 24	23 16	15 8	7 0
S0	SFT[1:0] SFEC[2:0]	SFID1[10:0]	Res.	SFID2[10:0]

Table 404. 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.

49.3.9 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 405. 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 406. 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.

49.4 FDCAN registers

49.4.1 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

49.4.2 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.

49.4.3 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.

49.4.4 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.

49.4.5 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.

49.4.6 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.

49.4.7 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.

49.4.8 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.

49.4.9 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.

49.4.10 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*.

49.4.11 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.

49.4.12 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.

49.4.13 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.

49.4.14 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.

49.4.15 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

49.4.16 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

49.4.17 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

49.4.18 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

49.4.19 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 547* and *Figure 548*.

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.

49.4.20 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.

49.4.21 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.

49.4.22 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.

49.4.23 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].

49.4.24 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.

49.4.25 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].

49.4.26 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.

49.4.27 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).

49.4.28 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.

49.4.29 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.

49.4.30 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

49.4.31 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

49.4.32 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

49.4.33 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

49.4.34 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

49.4.35 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.

49.4.36 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].

49.4.37 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.

49.4.38 FDCAN register map

Table 407. 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 407. 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	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.	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	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 407. 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	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 2.3 on page 87* for the register boundary addresses.

50 Universal serial bus full-speed device interface (USB)

50.1 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.

50.2 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)

50.3 USB implementation

Table 408 describes the USB implementation in the devices.

Table 408. STM32L552xx and STM32L562xx USB implementation

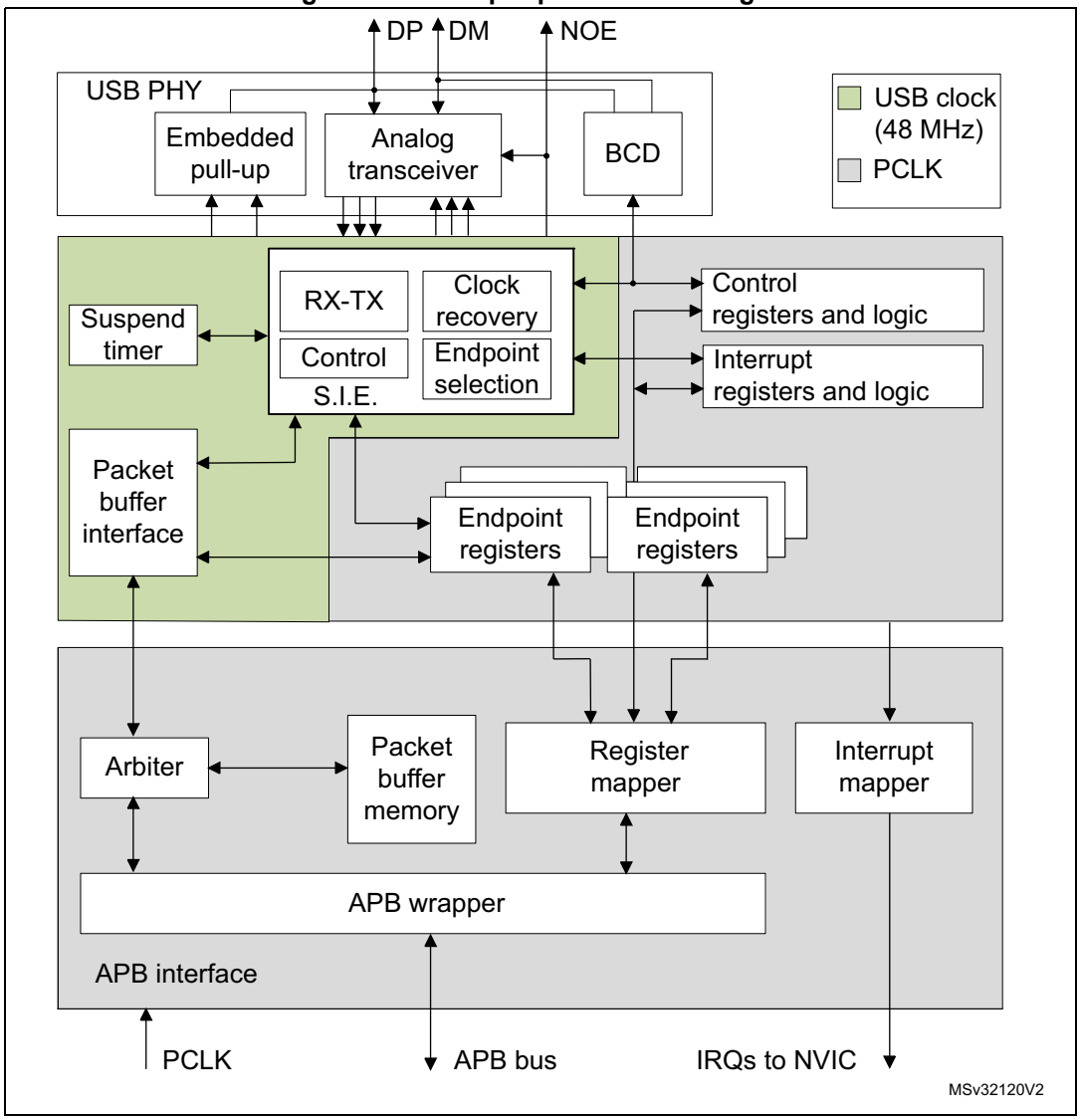
USB features(1)	USB
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	1024 bytes
Dedicated packet buffer memory SRAM access scheme	2 x 16 bits / word
USB 2.0 Link Power Management (LPM) support	X
Battery Charging Detection (BCD) support	X
Embedded pull-up resistor on USB_DP line	X

1. X= supported

50.4 USB functional description

Figure 549 shows the block diagram of the USB peripheral.

Figure 549. 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.

50.4.1 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.

50.5 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.

50.5.1 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.

50.5.2 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

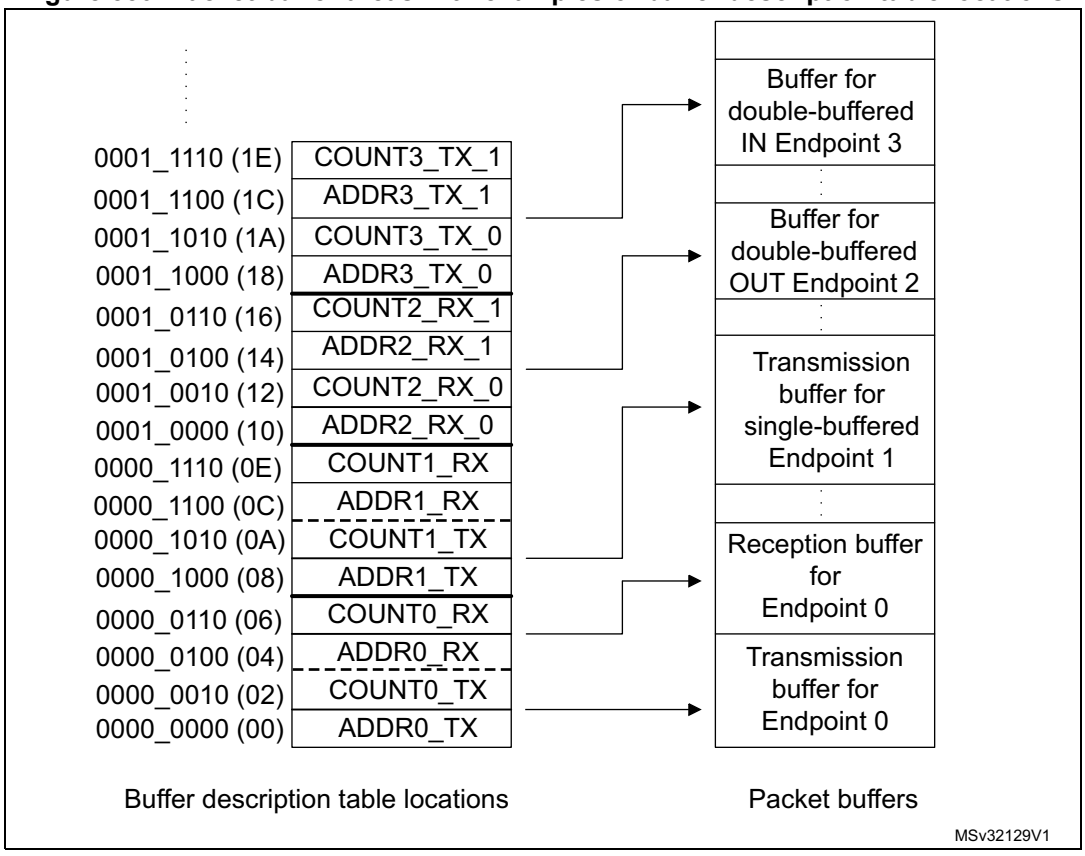
Each bidirectional endpoint may receive or transmit data from/to the host. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface 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: Due to USB data rate and packet memory interface requirements, the APB1 clock must have a minimum frequency of 10 MHz to avoid data overrun/underrun problems.

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 50.6.2: 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 50.5.4: Isochronous transfers* and *Section 50.5.3: Double-buffered endpoints* respectively). The relationship between buffer description table entries and packet buffer areas is depicted in *Figure 550*.

Figure 550. 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 1952*) 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.

50.5.3 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 409. 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 410. 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 410 on page 1958*). 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.

50.5.4 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 411*.

Table 411. 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.

50.5.5 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. Even if this event can trigger an WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. 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.
2. Clear FSUSP bit of USB_CNTR register.
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 412*, 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 412. 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 412. 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.

50.6 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 2.3: 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 76* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

50.6.1 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 1952*).

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 50.5.3: 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 50.5.4: 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 413: Reception status encoding on page 1974*.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 50.5.3: 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 414: Endpoint type encoding on page 1974*. 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 50.5.4: 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 415* 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 50.5.3: 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 50.5.3: 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 50.5.4: 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 416*. 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 50.5.3: 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 413. 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 414. Endpoint type encoding

EP_TYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 415. 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 416. 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.

50.6.2 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 1952*.

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 417*.

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 417. 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

50.6.3 USB register map

The table below provides the USB register map and reset values.

Table 418. 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 418. 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 2.3 on page 87* for the register boundary addresses.

51 USB Type-C™ / USB Power Delivery interface (UCPD)

51.1 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.

51.2 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

51.3 UCPD implementation

The devices have one UCPD controller to support one USB Type-C port.

Table 419. 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

51.4 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.

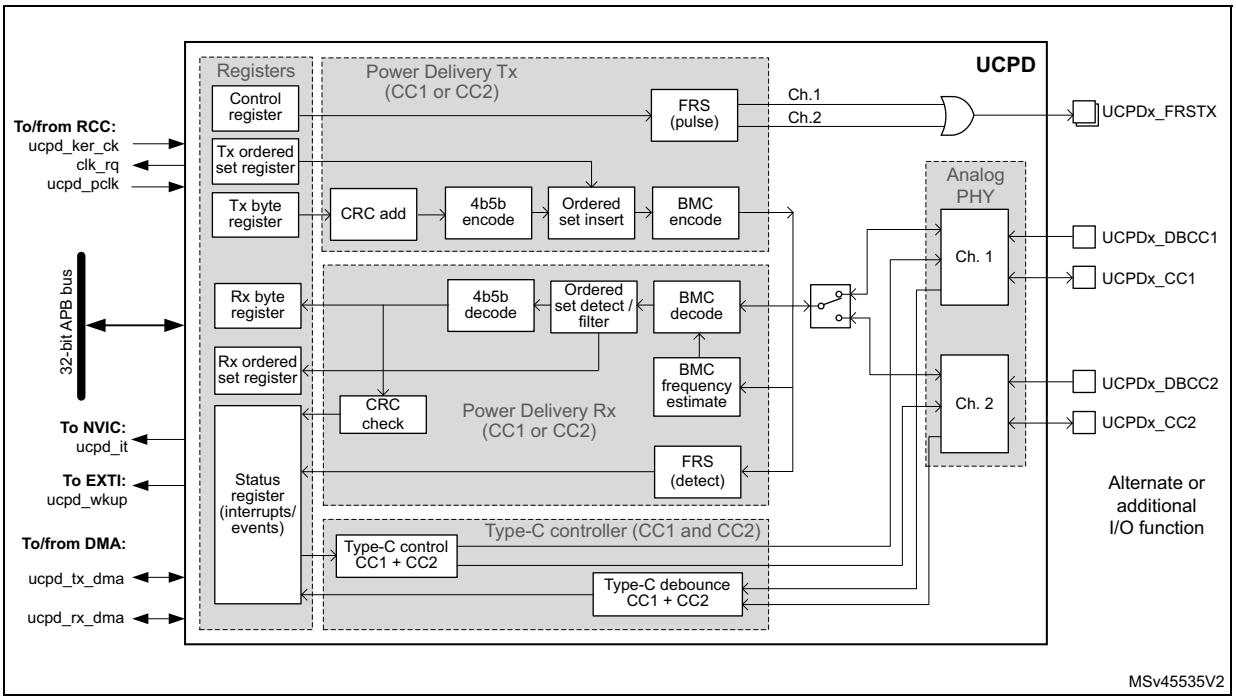
The UCPD transmitter BMC (bi-phase mark) encodes and transmits data: preamble, SOP, payload data from protocol layer (after 4b5b-encoding), CRC, and EOP on the Type-C connector CC lines. It automatically inserts inter-frame gap and executes “Hard Reset”.

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.

51.4.1 UCPD block diagram

Figure 551. UCPD block diagram



The following table lists external signals (alternate or additional I/O functions).

Table 420. 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 421. 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

51.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

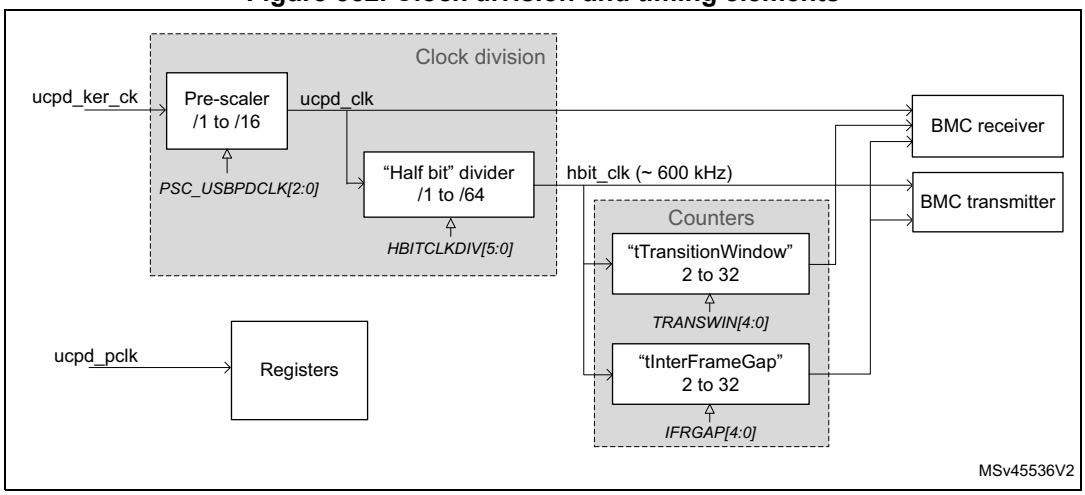
The register section is clocked with the APB clock (ucpd_pclk).

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 552. 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.

51.4.3 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 422. 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

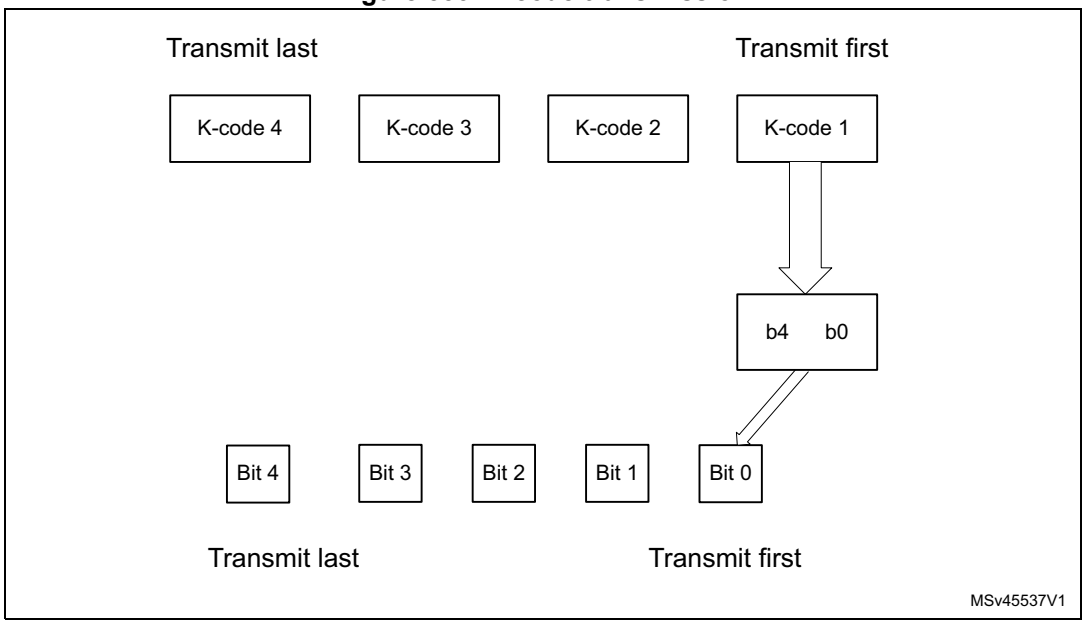
Table 422. 4b5b Symbol Encoding Table (continued)

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 553. 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 423. 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 424. 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 424. 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

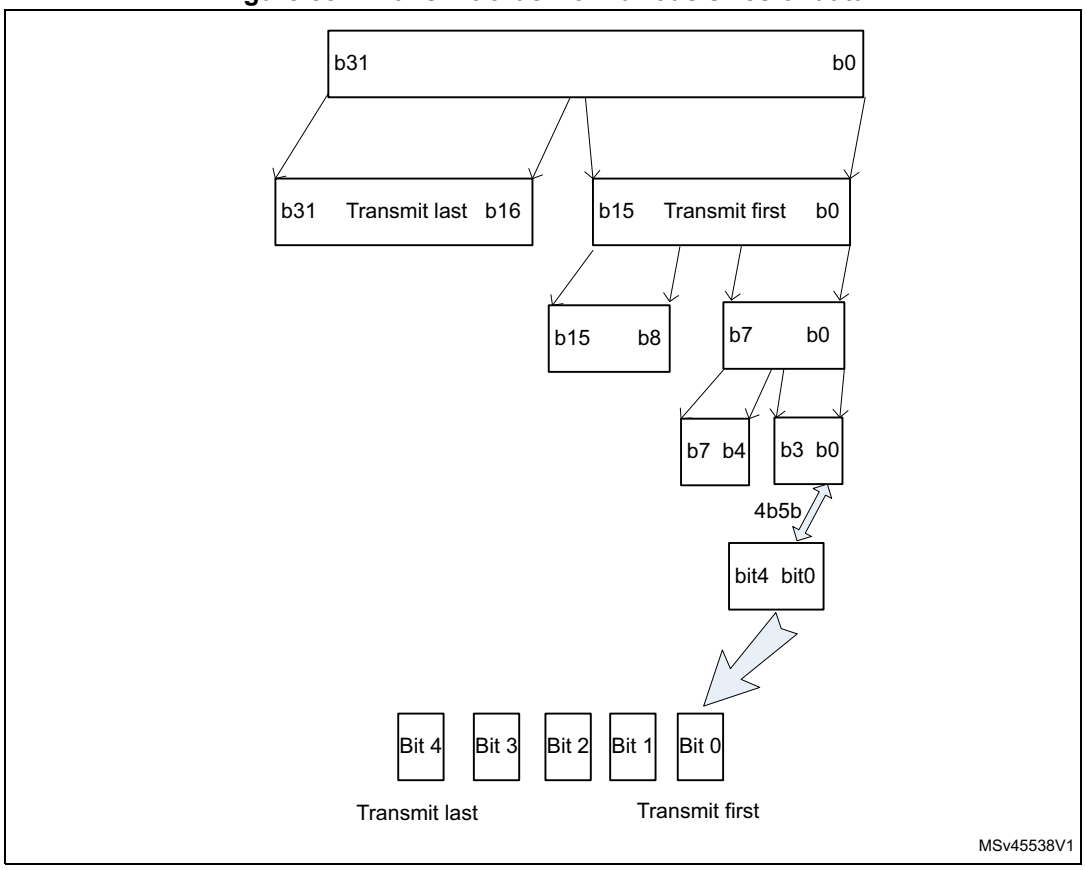
Allowed transmission data units / data sizes are in the following table.

Table 425. 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 554. Transmit order for various sizes of data

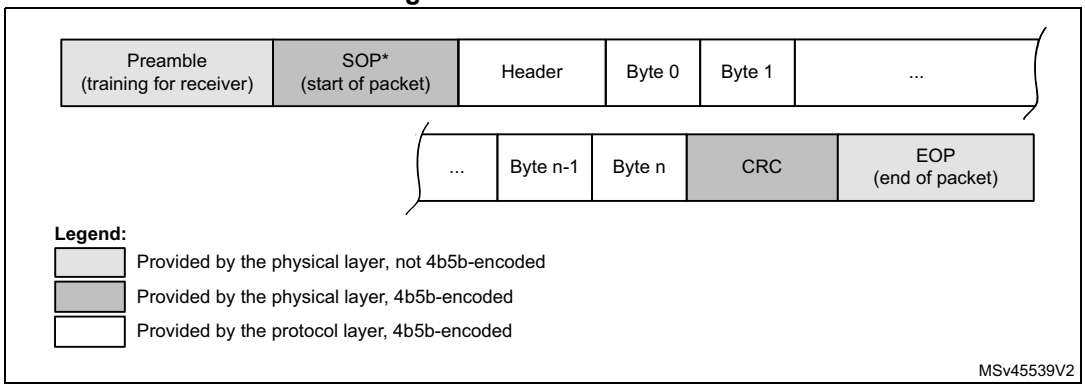


Packet format

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 555. Packet format



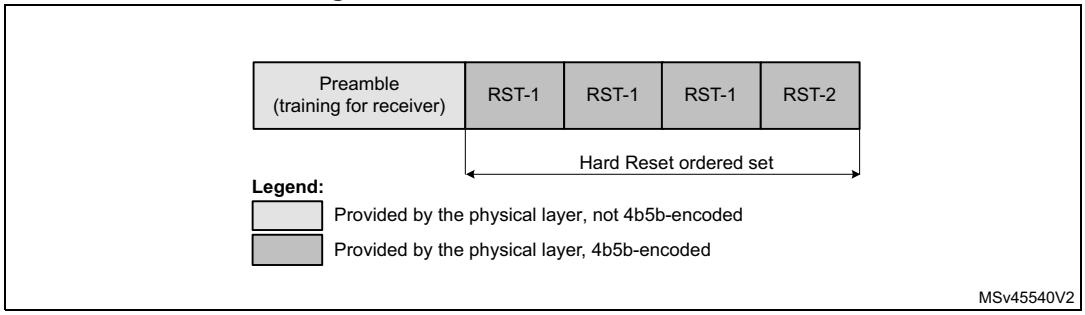
Hard Reset

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.

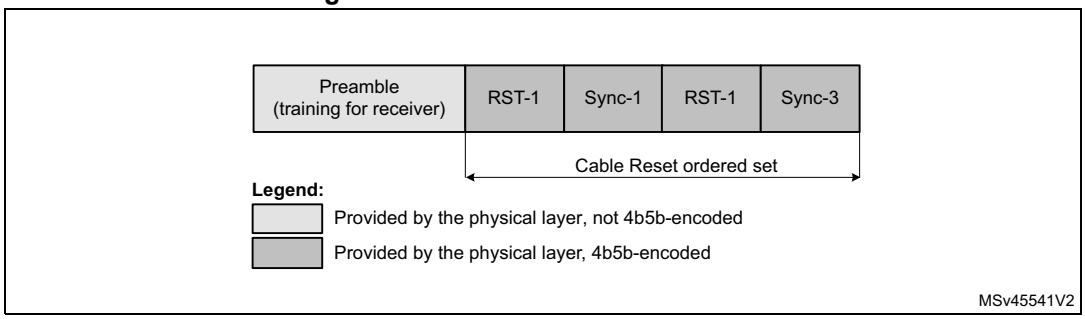
Figure 556. Line format of Hard Reset



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 557. 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:

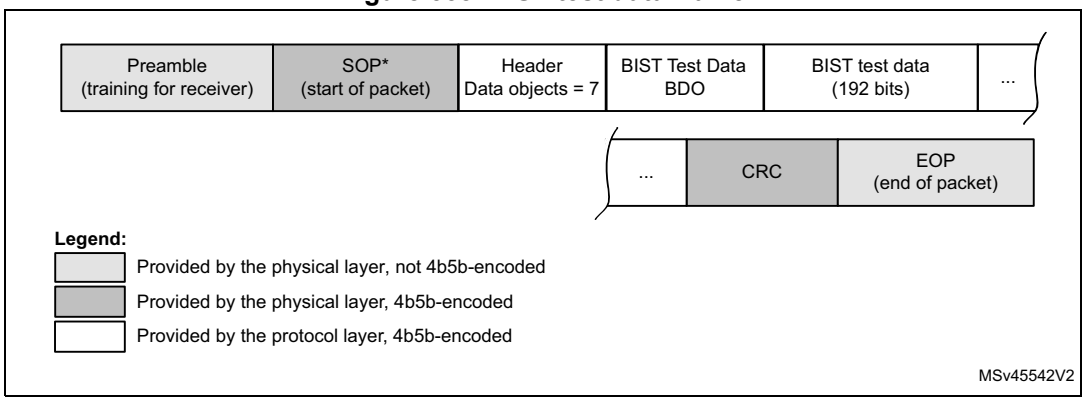
- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), 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 558. BIST test data frame



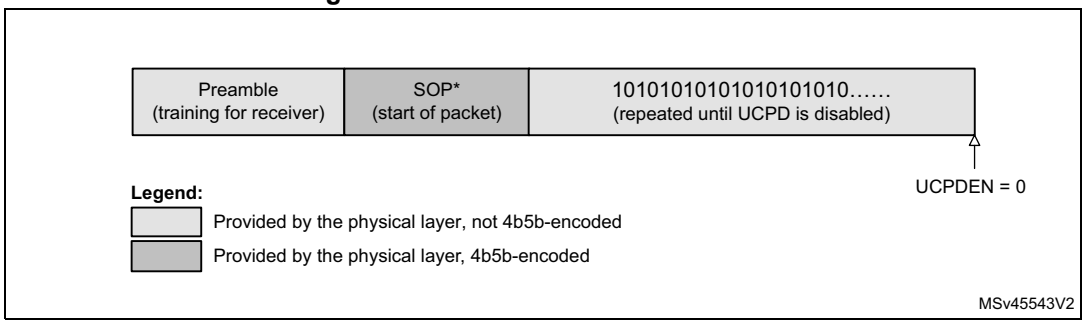
This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 555: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), 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.

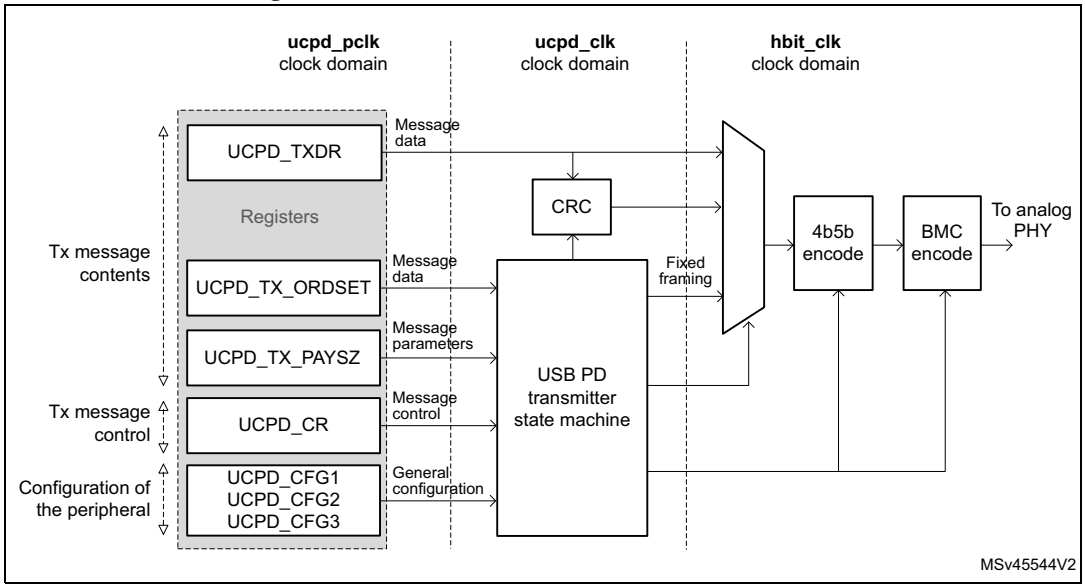
Figure 559. BIST Carrier Mode 2 frame



51.4.4 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 560. 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.

51.4.5 UCPD BMC receiver

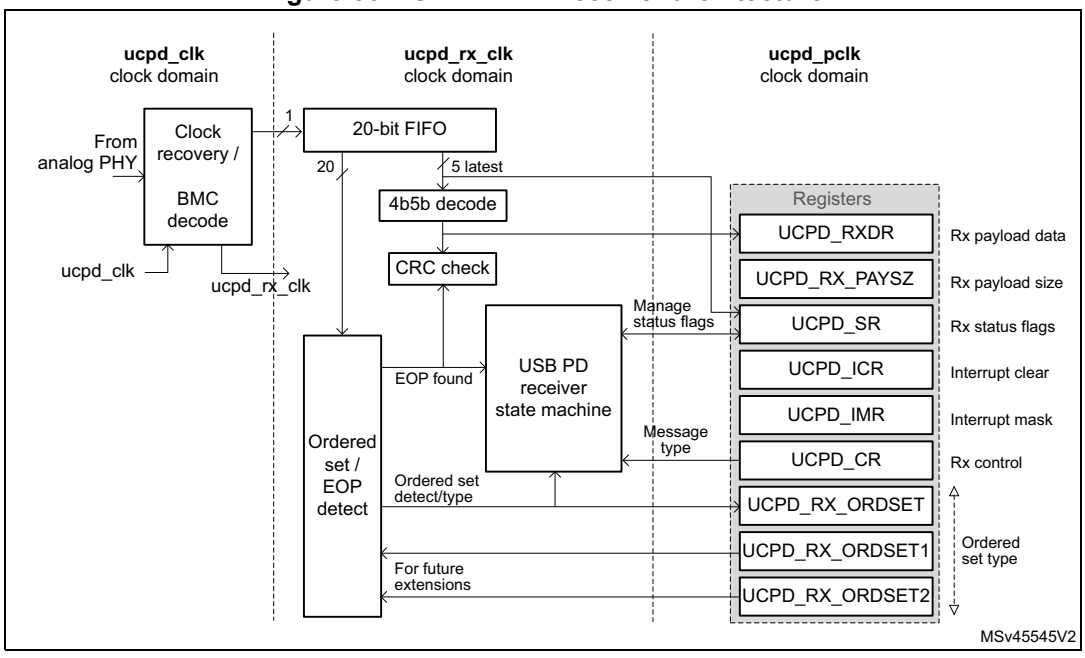
The UCPD BMC receiver performs:

- 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 561. 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.

51.4.6 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.

51.4.7 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.

51.4.8 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.

51.4.9 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.

51.4.10 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

51.4.11 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 426. 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 427. 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	Event => go to next line	Comments
Unattached. SRC	0:Source; 11:Rp3A0	11:both enabled	0 (don't care)	0: [Normal]	00: [neither]	PHYEVT 1: [VRd-3A0]	Wait for sink attach detect ; seen on CC1 [EVT1]
Attachwait. SRC			0 [Rd on CC1]			PHYEVT 2: [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)
						PHYEVT 1: [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]

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.

51.5 UCPD low-power modes

A summary of low-power modes is shown below in *Table 428*.

Table 428. 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. Set the STDBY bit of the PWR_CR3 register just before entering Standby mode, and clear it upon exiting the Standby mode as the very first action on UCPD.
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)

51.6 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

Table 429. 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.

51.7 UCPD registers

51.7.1 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

51.7.2 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).

51.7.3 UCPD configuration register 3 (UCPD_CFGR3)

Address offset: 0x008

Reset value: 0x0000 0000

Configuration of UCPD analog PHY trimming. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0). The trim values of all resistors are determined by hardware until the first software write into the register is performed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TRIM2_NG_CC3A0[3:0]				TRIM2_NG_CC1A5[4:0]					TRIM2_NG_CCRPD[3:0]			
			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
Res.	Res.	Res.	TRIM1_NG_CC3A0[3:0]				TRIM1_NG_CC1A5[4:0]					TRIM1_NG_CCRPD[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 28:25 **TRIM2_NG_CC3A0[3:0]**: SW trim value for RP3A0 resistors on the CC2 line

Bits 24:20 **TRIM2_NG_CC1A5[4:0]**: SW trim value for RP1A5 resistors on the CC2 line

Bits 19:16 **TRIM2_NG_CCRPD[3:0]**: SW trim value for RPD resistors on the CC2 line

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:9 **TRIM1_NG_CC3A0[3:0]**: SW trim value for RP3A0 resistors on the CC1 line

Bits 8:4 **TRIM1_NG_CC1A5[4:0]**: SW trim value for RP1A5 resistors on the CC1 line

Bits 3:0 **TRIM1_NG_CCRPD[3:0]**: SW trim value for RPD resistors on the CC1 line

51.7.4 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 426: 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 426: 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).

51.7.5 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.
											rw				

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

51.7.6 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

51.7.7 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.

51.7.8 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.

51.7.9 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)

51.7.10 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

51.7.11 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

51.7.12 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).

51.7.13 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

51.7.14 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.

51.7.15 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.

51.7.16 UCPD register map

Table 430. 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_CFG1**	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_CFG2**	Res.	Res.	Res.	Res.	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	UCPD_CFG3	Res.	Res.	Res.	TRIM2_NG_CC3A0[3:0]				TRIM2_NG_CC1A5[4:0]					TRIM2_NG_CCRPD[3:0]				Res.	Res.	Res.	TRIM1_NG_CC3A0[3:0]				TRIM1_NG_CC1A5[4:0]					TRIM1_NG_CCRPD[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

Table 430. 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
0x00C	**UCPD_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN	DBATTEN	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	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_ORDSET**	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_PAYSZ**	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_ORDSET**	Res.	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_PAYSZ**	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

Table 430. 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
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_ ORDEXT1**	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
0x038	**UCPD_RX_ ORDEXT2**	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 2.3 on page 87* for the register boundary addresses.

52 Debug support (DBG)

52.1 Introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core
- Code execution tracing
- Software instrumentation
- Cross-triggering

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm® Coresight™ components.

- SWJ-DP: JTAG/Serial-wire debug port
- AHB-AP: AHB access port
- ROM table
- System Control Space (SCS)
- Breakpoint Unit (BPU)
- Data Watchpoint and Trace Unit (DWT)
- Instrumentation Trace Macrocell (ITM)
- Embedded Trace Macrocell (ETM)
- Cross Trigger Interface (CTI)
- Trace Port Interface Unit (TPU)

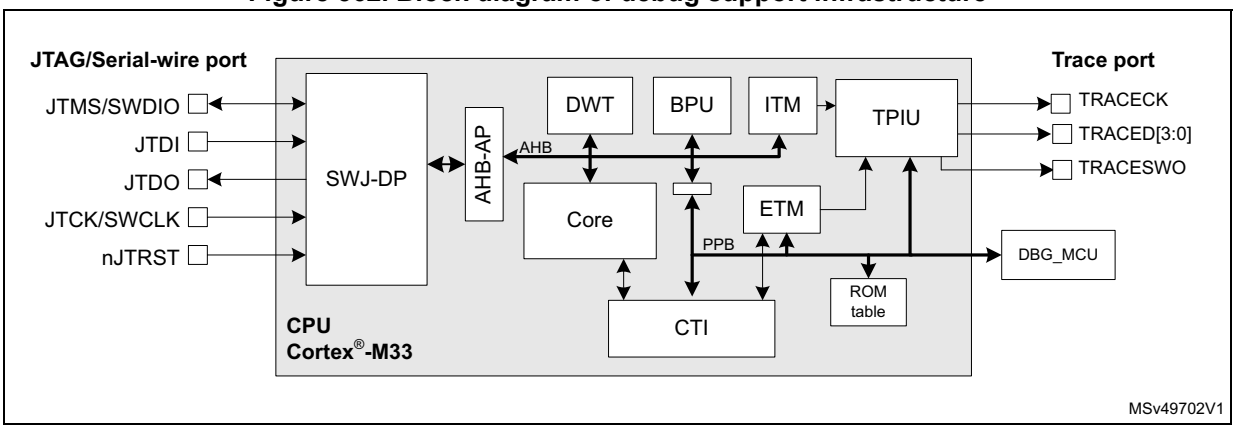
The debug features are accessible by the debugger via the AHB-AP.

Additional information can be found in the Arm® documents referenced in *Section 52.12*.

52.2 DBG functional description

52.2.1 DBG block diagram

Figure 562. Block diagram of debug support infrastructure



52.2.2 DBG pins and internal signals

Table 431. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	Serial wire data in/out	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial wire clock	PA14
JTDI(1)	I	JTAG test data input	-	-	PA15
JTDO	O	JTAG test data output	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

1. TDI is hosted on the same IO as a USBPD-CC line. To avoid pull-up/down conflict, a user option allows to decide whether the pad is used as TDI or as CC.

Table 432. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED0	O	Trace synchronous data out 0	Refer to datasheet
TRACED1		Trace synchronous data out 1	
TRACED2		Trace synchronous data out 2	
TRACED3		Trace synchronous data out 3	
TRACECK		Trace clock	

Table 433. Single Wire Trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Single-wire trace asynchronous data out	PB3(1)

1. TRACESWO is multiplexed with JTDO. This means that single-wire trace is only available when using the Serial-wire debug interface, and not when using JTAG.

52.2.3 DBG reset and clocks

The debug port (SWJ-DP) is reset by a power-on reset and when waking up from Standby mode.

The debugger supplies the clock for the debug port via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both Serial-wire and JTAG modes, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DCLK domain, which covers the rest of the SWJ-DP and the access port.

The DCLK is a gated version of the system clock.

The DCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the debug port CTRL/STAT register. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in the debug port CTRL/STAT register. The DCLK is disabled at power-up, and must be disabled when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor are clocked with the processor clock.

52.2.4 DBG power domains

The debug components are located in the core power domain. This means that debugger connection is not possible in shutdown or standby low-power modes. To avoid losing the connection when the device enters standby mode, it is possible to maintain the power to the core by setting a bit in the DBGMCU_CR register. This also keeps the processor clocks active, and holds off the reset, so that the debug session is maintained.

52.2.5 Debug and low-power modes

The STM32L552xx and STM32L562xx devices include power saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off, or the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power saving mode emulation has been implemented. If emulation is enabled for a domain, the domain still enters power saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power saving mode, but the debugger does not lose the connection.

Emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information refer to *Section 52.11: Microcontroller debug unit (DBGMCU)*.

52.2.6 Security

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features can not be used to alter or compromise the normal operation of the finished product, these features can be disabled, or limited in scope. For example, secure software debug and trace can be disabled without preventing the debug of non-secure code.

There are four authentication signals used by the system to determine which debug features are enabled or disabled. These signals are:

- **dbgen**: global enable for all debug features
  0: all debug features are disabled.
  1: debug features in non-secure state are enabled. Debug features in secure state are dependent on the state of the **spiden** signal.
- **spiden**: enables debug in secure state when **dbgen** = 1
  0: debug features are disabled in secure state.
  1: debug features are enabled in secure state.
- **niden**: enables trace and performance monitoring (non-invasive debug)
  0: trace generation is disabled.
  1: trace generation in non-secure state is enabled. Trace generation in secure state is dependent on the state of the **spniden** signal.
- **spniden**: enables trace and performance monitoring in secure state when **niden** = 1.
  0: trace generation is disabled in secure state.
  1: trace generation is enabled in secure state.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component chapter, or to the relevant Arm® technical documentation.

The state of the signals are set according to the read data protection (RDP) level (see *Section 6.7.2: Readout protection (RDP)*), as shown in *Table 434*:

Table 434. Authentication signal states

RDP level	Authentication signal state	Description
0	DBGEN = 1 SPIDEN = 1 NIDEN = 1 SPNIDEN = 1	Debug and trace is enabled whatever the state of the processor. Debugger access to secure memory is permitted.
0.5	DBGEN = 1 SPIDEN = 0 NIDEN = 1 SPNIDEN = 0	Debug and trace is enabled when the processor is in non-secure state. Debugger access to secure memory is disabled.

Table 434. Authentication signal states (continued)

RDP level	Authentication signal state	Description
1	DBGEN = 1 SPIDEN = 0 NIDEN = 1 SPNIDEN = 0	Debug and trace is enabled when the processor is in non-secure state. Debugger access to secure memory is disabled, as well as to the following areas: Flash memory, SRAM2, Backup registers, ICACHE, on-the-fly decryption region (OCTOSPI).
2	DBGEN = 0 SPIDEN = 0 NIDEN = 0 SPNIDEN = 0	Debug and trace is disabled.

Note: Security features are only relevant when option bit TZEN = 1. If security features are disabled, the authentication signals are still set according to the RDP level, but since the processor and all memory are non-secure, SPNIDEN and SPIDEN are redundant.

The state of the authentication signals can be read from the DAUTHSTATUS register in the system control space (SCS) of the Cortex-M33.

Note that debugger access to secure memory (when permitted) must be performed using secure transactions on the debug AHB, that is, with the PROT[6] bit set in the AP_CSWR register.

Debugger access is disabled while the processor is booting from system flash (RSS), whatever the RDP level, if security features are enabled (TZEN = 1).

52.2.7 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a Coresight™ component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- a 5-pin standard JTAG interface (JTAG-DP)
- a 2-pin (clock + data) Serial-wire debug port (SW-DP)

The two modes are mutually exclusive, since they share the same IO pins.

By default the JTAG-DP is selected after a system or a power-on reset. The five IO pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

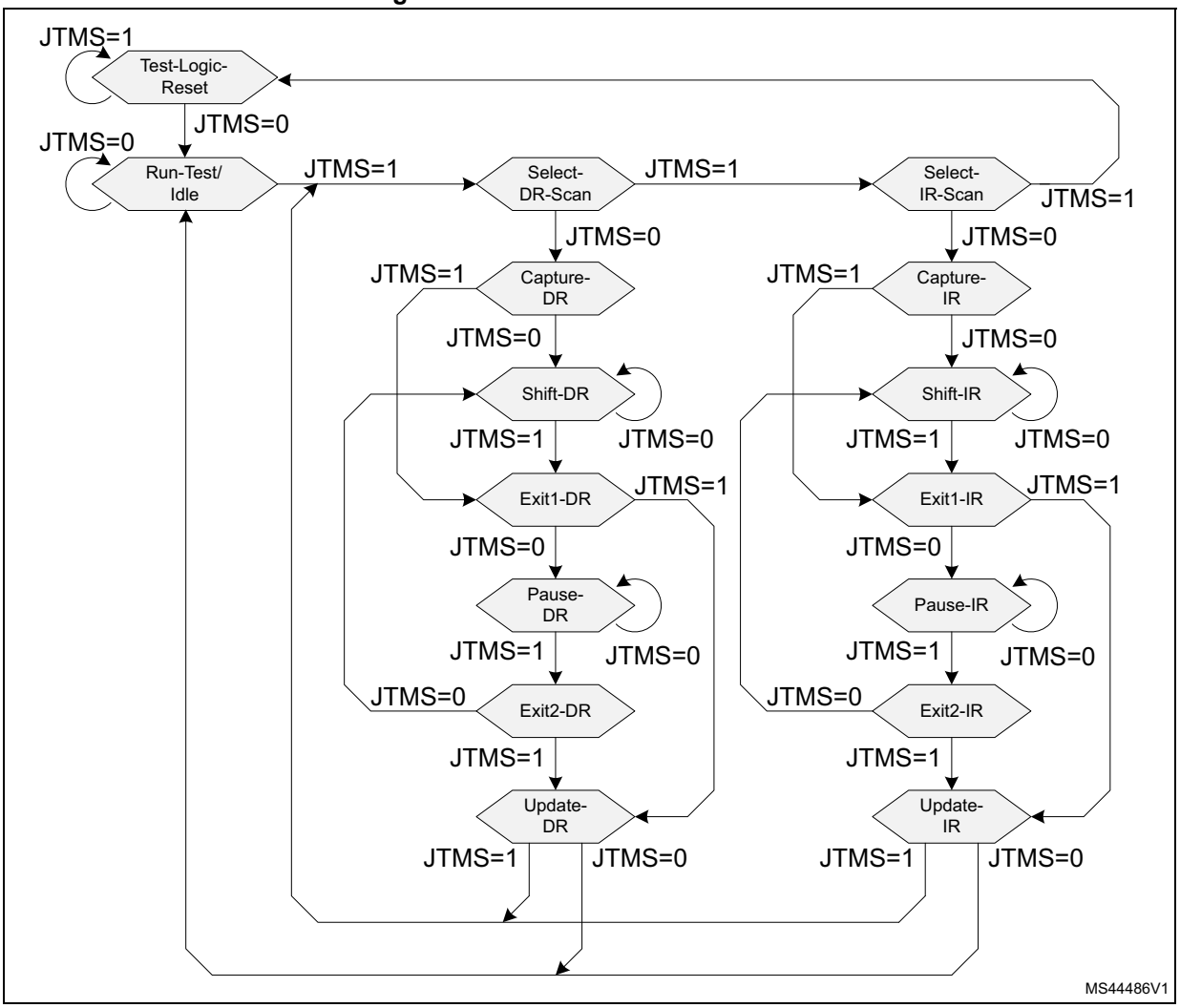
JTCK/SWCLK must be cycled for each data bit.

In SW-DP mode, the unused JTAG pins JTDI, JTDO and nJTRST can be used for other functions. Note that all SWJ port IOs can be reconfigured to other functions by software, but debugging is no longer possible.

52.2.8 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM), shown in *Figure 563*, based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR), and the other one with a number of data registers (DR).

Figure 563. JTAG TAP state machine



The operation of the JTAG-DP is as follows:

1. When the TAPSM goes through the Capture-IR state, 0b0001 is transferred onto the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
2. While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
   – The LSB of the IR scan chain is output on JTDO.
   – Bit[n] of the IR scan chain is transferred to bit[n-1].
   – The value on JTDI is transferred to the MSB of the IR scan chain.
3. When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the instruction register.
4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers onto one of the DR scan chains, connected between JTDI and JTDO.
5. The value held in the instruction register determines which data register, and associated DR scan chain, is selected.
6. This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shift in the Shift-IR state.
7. When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected data register.
8. When the TAPSM is in the Run-Test/Idle state, no special actions occurs. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the test-logic-reset state.

The data registers corresponding to the 4-bit IR instructions are listed in *Table 435*.

Table 435. JTAG-DP data registers

IR instruction	DR register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – bits 31:1 = reserved – bit 0 = APABORT: write 1 to generate an AP abort.
1001	(BYPASS)	1	Reserved: BYPASS selected
1010	DPACC	35	Debug port access register Initiates the debug port and gives access to a debug port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request bits 2:1 = A[3:2] = 2-bit address of a debug port register bit 0 = RnW = read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit Acknowledge: – 010 = OK/FAULT – 001 = WAIT – others = reserved

Table 435. JTAG-DP data registers (continued)

IR instruction	DR register	Scan chain length	Description
1011	APACC	35	Access port access register Initiates an access port and gives access to an access port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request bits 2:1 = A[3:2] = 2-bit sub-address of an access port register bit 0 = RnW= Read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit Acknowledge: – 010 = OK/FAULT – 001 = WAIT – others= reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected
1110	IDCODE	32	Identification code 0x0BA0 4477: Cortex®-M33 JTAG debug port ID code
1111	BYPASS	1	Bypass A single JTCK cycle delay is inserted between JTDI and JTDO.

The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification *[1]*.

52.2.9 Serial-wire debug port

The Serial-wire debug protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock.

A transfer comprises three phases:

1. packet request (8 bits) transmitted by the host (see *Table 436*)
2. acknowledge response (3 bits) transmitted by the target (see *Table 437*)
3. data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read) (see *Table 438*)

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

Table 436. Packet request

Bit field	Name	Description
0	Start	Must be 1.
1	APnDP	– 0: DP register access - see *Section 52.2.10: Debug port registers* – 1: AP register access - see *Section 52.3: Access ports*
2	RnW	– 0: write request – 1: read request
4:3	A(3:2)	Address field of the DP or AP register (refer to *Table 439* or *Table 440*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as 1 by target.

Table 437. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 438. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or read data
32	Parity	Single bit parity of 32 data bits

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial-wire debug protocol, refer to the Arm® Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

52.2.10 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in *Table 439*.

The debugger can access the DP registers as follows:

1. Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the data field with the write data. If using SWD, the A(3:2) and RnW fields are

part of the packet request word sent to the SW-DP with the APnDP bit reset (see *Table 436*). The write data are sent in the data phase.

2. To access one of the banked DP registers at address 0x4, the register number must first be written to the DP_SELECTR register at address 0x8. Any subsequent read or write to address 0x4 access the register corresponding to the contents of the DP_SELECTR register.

DP debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x0BE0 2477 (SW-DP), 0x0BE0 1477 (JTAG-DP)

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

Bits 31:28 **REVISION[3:0]**: revision code
0x0 (JTAG-DP): r0p0
0x0 (SW-DP): r0p0

Bits 27:20 **PARTNO[7:0]**: part number for the debug port
0xBE

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

Bit 16 **MIN**: minimal debug port (MINDP) implementation
0x1: MINDP implemented (transaction counter and pushed operations are not supported)

Bits 15:12 **VERSION[3:0]**: debug port architecture version
0x1 (JTAG-DP): DPv1
0x2 (SW-DP): DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

DP abort register (DP_ABORTR)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
											w	w	w		w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: overrun error clear
0: no effect
1: STICKYORUN bit cleared in DP_CTRL/STATR register

Bit 3 **WDERRCLR**: write data error clear
0: no effect
1: WDATAERR bit cleared in DP_CTRL/STATR register

Bit 2 **STKERRCLR**: sticky error clear
0: no effect
1: STICKYERR bit cleared in DP_CTRL/STATR register

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DAPABORT**: current AP transaction aborted if excessive number of WAIT responses returned
This bit indicates that the transaction is stalled.
0: no effect
1: transaction aborted

DP control and status register (DP_CTRL/STATR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		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.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
								r	r	r				r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGPWRUPACK**: debug power-up acknowledge
See description in *Section 52.2.5: Debug and low-power modes*.
0: DCLK gated
1: DCLK enabled

Bit 28 **CDBGPWRUPREQ**: debug power-up request
Controls the DCLK enable request signal.
0: requests DCLK gating
1: requests DCLK enable

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **WDATAERR**: write data error (read-only) in SW-DP
Indicates that there is a parity or framing error on the data phase of a write
or
a write accepted by the DP is then discarded without being submitted to the AP.
This bit is reset by writing 1 to the ABORT.WDERRCLR bit.
0: no error
1: an error occurred
Note: this bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response (read-only) in SW-DP.
Indicates the response to the last AP read access.
0: read not OK
1: read OK
Note: this bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: transaction error (read-only in SW-DP, read/write in JTAG-DP)
Indicates that an error occurred in an AP transaction.
This bit is reset by writing 1 to the DP_ABORTR.STKERRCLR bit (in SW-DP and JTAG-DP)
0: no error
1: an error occurred

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STICKYORUN**: overrun (read-only in SW-DP, read/write in JTAG-DP).
Indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.
This bit is reset by writing 1 to the DP_ABORTR.ORUNERRCLR bit (in SW-DP and JTAG-DP)
0: no overrun
1: an overrun occurred

Bit 0 **ORUNDETECT**: overrun detection mode enable.
0: disabled
1: enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

DP data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x1.

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

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

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**: tristate period for SWDIO
0x0: 1 data bit period

Bits 7:6 **WIREMODE[1:0]**: SW-DP mode
0x0: synchronous mode

Bits 5:0 Reserved, must be kept at reset value.

DP target identification register (DP_TARGETIDR)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x2.

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

Bits 31:28 **TREVISION[3:0]**: target revision
0x1: revision 1

Bits 27:12 **TPARTNO[15:0]**: target part number
0x0472: STM32L552xx and STM32L562xx

Bits 11:1 **TDESIGNER[10:0]**: target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x3.

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

Bits 31:28 **TINSTANCE[3:0]**: target instance number
Defines the instance number for the device in a multi-drop system.
0x0: instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial-wire debug protocol version
0x1: version 2

DP event status (DP_EVENTSTATR)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x4.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EA**: event status flag
0: Cortex®-M33 processor halted
1: Cortex®-M33 processor not halted

DP event status register (DP_RESENDR)

Address offset: 0x8

Reset value: 0x0000 0000

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

Bits 31:0 **RESEND[31:0]**: value returned by the last AP read or DP_RDBUFF read
This register is used in the event of a corrupted read transfer.

DP access port select register (DP_SELECTR)

Address offset: 0x8

Reset value: 0xXXXX XXXX

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

Bits 31:28 **APSEL[3:0]**: access port select
Selects the access port for the next transaction.
0x0: AP0 - Cortex®-M33 debug access port (AHB-AP)
others: reserved

Bits 27:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select.
Selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select
Selects the register at address 0x4 of the debug port.
0x0: DP_CTRL/STAT register
0x1: DP_DLCR register
0x2: DP_TARGETID register
0x3: DP_DLPIDR register
0x4: DP_EVENTSTAT register
others: reserved

DP read buffer register (DP_RDBUFFR)

Address offset: 0xC

Reset value: 0x0000 0000

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

Bits 31:0 **RDBUFF[31:0]**: value returned by the last AP read access
The value returned by an AP read access can either be obtained using a second read access to the same address, that initiates a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.

52.2.11 Debug port register map and reset values

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interface.

The debug port address is 2-bit wide, defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 439. Debug port 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	DP_DPIDR	REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION[3:0]				DESIGNER[10:0]											Res.
	Reset value	0	0	0	0	1	0	1	1	1	1	1	0				0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	1
0x0	DP_ABORTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	STKCMPCLR	DAPABORT
	Reset value																												0	0	0	0	0
0x4(1)	DP_CTRL/STATR	Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
	Reset value			0	0																					0	0	0				0	0
0x4(2)	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0	0	0						
0x4(3)	DP_TARGETIDR	TREVISION[3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	
0x4(4)	DP_DLPIDR	TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
	Reset value	0	0	0	0																								0	0	0	1	
0x4(5)	DP_EVENTSTATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EA
	Reset value																																1
0x8	DP_RESENDR	RESEND[0:31]																															
	Reset value	0	0	0	0	0	0	0	0	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 439. Debug port 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
0x8	DP_SELECTR	APSEL[0:3]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
	Reset value	x	x	x	x																					x	x	x	x	x	x	x	x
0xC	DP_RDBUFFR	RDBUFF[0:31]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DP_SELECTR.DPBANKSEL[3:0] = 0x0.
2. DP_SELECTR.DPBANKSEL[3:0] = 0x1.
3. DP_SELECTR.DPBANKSEL[3:0] = 0x2.
4. DP_SELECTR.DPBANKSEL[3:0] = 0x3.
5. DP_SELECTR.DPBANKSEL[3:0] = 0x4.

52.3 Access ports

There is one access port (AP) attached to the DP. It enables access to the debug and trace features integrated in the Cortex®-M33 processor core via its internal AHB bus.

52.3.1 Access port registers

The access port is of type MEM-AP, that is to say the debug and trace component registers are mapped in the address space of the AHB. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 440*.

The address of the AP registers is composed of the following fields:

- bits [7:4]: content of the DP_SELECTR.APBANKSEL[3:0] field in the DP (see *DP access port select register (DP_SELECTR)*)
- bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 435*), or of the SW-DP packet request (see *Table 436*), depending on the debug interface used
- bits [1:0]: always set to 0

The content of the DP_SELECTR.APSEL[3:0] field in the DP defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program the DP_SELECTR.APSEL[3:0] field in the DP to choose the AP, and the APBANKSEL[3:0] field to select the register bank to be accessed (see *DP access port select register (DP_SELECTR)*).
2. Program the A(3:2) field in the APACC data register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW

fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see *Table 436*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the TAR register.
2. Program the AP_CSWR register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the AP_DRWR register to initiate a bus transaction at the address held in the AP_TAR register. Alternatively, a read or write to banked data register AP_BDnR triggers an access to address TAR[31:4] + n, allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register.

For more detailed information on the MEM-AP refer to the Arm® Debug Interface Architecture Specification *[1]*.

AP control/status word register (AP_CSWR)

Address offset: 0x0

Reset value: 0x0100 00X0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PROT[6]	Res.	Res.	PROT[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw			rw	rw	rw	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.	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
									r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PROT[6]**: secure transfer request
This field sets the protection attribute HPROT[6] of the bus transfer.
0: secure transfer
1: non-secure transfer

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

Bits 27:24 **PROT[3:0]**: bus transfer protection
This field sets the protection attributes HPROT[3:0] of the bus transfer.
0bXXX1: data access (bit 24 is read only)
0bXX0X: non-privilege mode
0bXX1X: privilege mode
0bX0XX: non-bufferable
0bX1XX: bufferable
0b0XXX: non-shareable, no look-up, non-modifiable
0b1XXX: shareable, look-up, modifiable

Bits 23:7 Reserved, must be kept at reset value.

Bit 6 **DBGSTATUS**: device enable (DEVICEEN) status
0: AHB transfers blocked
1: AHB transfers enabled

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size in bytes of the transaction (SIZE field)
other: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction
0x0: byte (8-bit)
0x1: halfword (16-bit)
0x2: word (32-bit)
others: reserved

AP transfer address register (AP_TAR)

Address offset: 0x04

Reset value: 0xXXXX XXXX

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

Bits 31:0 **TA[31:0]**: address of current transfer

AP data read/write register (AP_DRWR)

Address offset: 0x0C

Reset value: 0xXXXX XXXX

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

Bits 31:0 **TD[31:0]**: data of current transfer

AP banked data n register (AP_BDnR)

Address offset: 0x10 + 4 * n, (n = 0 to 3)

Reset value: 0x0000 0000

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

Bits 31:0 **TBD[31:0]:** banked data of current transfer to address TAR
TA + AP_BDnR address [3:2] + 0b00.
The auto address incrementing is not performed on AP_BD0-3R. Banked transfers are only supported for word transfers.

AP configuration register (AP_CFGR)

Address offset: 0xF4

Reset value: 0x0000 0000

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

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LD**: large data
0: data not larger than 32-bits supported

Bit 1 **LA**: long address
0: Physical addresses not larger than 32-bits supported

Bit 0 **BE**: big endian
0: only little-endian supported

AP base address register (AP_BASER)

Address offset: 0xF8

Reset value: 0xE00F E003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[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
BASEADDR[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[31:12]**: base address (bits 31 to 12) of the first ROM table
The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
0xE00FE

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: base-address register format
1: Arm® debug interface v5

Bit 0 **ENTRYPRESENT**: debug components presence
Indicates that debug components are present on the access port bus.
1: debug components present

AP identification register (AP_IDR)

Address offset: 0xFC

Reset value: 0x1477 0015

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3]
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
CLASS[2:0]			Res.	Res.	Res.	Res.	Res.	VARIANT[3:0]				TYPE[3:0]			
r	r	r						r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: revision number
0x1: r0p1

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank.
0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code.
0x3B: Arm®

Bits 16:13 **CLASS[3:0]**:
0x8: MEM-AP

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **VARIANT[3:0]**:
0x1: Cortex®-M33

Bits 3:0 **TYPE[3:0]**:
0x5: AHB5

52.3.2 Access port register map and reset values

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interfaces.

The access port address is 8-bit wide, defined by DP_SELECTR.APBANKSEL[3:0] field and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 440. Access port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP_CSWR	Res.	PROT[6]	Res.	Res.	PROT[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		0			0	0	0	1																		X	X	X		0	0	0
0x04	AP_TAR	TA[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	Reserved	Reserved																															
0x0C	AP_DRWR	TD[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
0x10	AP_BD0R	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	AP_BD1R	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	AP_BD2R	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	AP_BD3R	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20 to 0xF0	Reserved	Reserved																															
0xF4	AP_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LD	LA	BE
	Reset value																														0	0	0
0xF8	AP_BASER	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT	
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0											1	1

Table 440. Access port 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
0xFC	AP_IDR	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	VARIANT[3:0]				TYPE[3:0]			
	Reset value	0	0	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	1	0	1	0	1

Refer to *Section 2.3* for the register boundary addresses.

52.4 ROM tables

The ROM table is a CoreSight™ component that contains the base addresses of all the Coresight debug components accessible via the AHB-AP. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There are two ROM tables in the CPU sub-system. The MCU ROM table is pointed to by the base register in the AHB-AP. It contains the base address pointer for the processor ROM table and for the TPIU registers.

The MCU ROM table (see *Table 441*) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F E000 to 0xE00F EFFC.

Table 441. MCU ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F E000	Processor ROM table	0xE00F F000	0x0000 1000	4	0x0000 1003
0xE00F E004	TPIU	0xE004 0000	0xFFF4 2000	4	0xFFF4 2003
0xE00F E008	Reserved	-	-	-	0x1FF0 2002
0xE00F E00C	Reserved	-	-	-	0x1FF0 2002
0xE00F F010	Top of table	-	-	-	0x0000 0000
0xE00F F014 to 0xE00F FFC8	Reserved	-	-	-	0x0000 0000
0xE00F FFCC to 0xE00F FFFC	ROM table registers	-	-	-	See *Table 443*

The processor ROM table contains the base-address pointer for the system control space (SCS) registers, that allow the debugger to identify the CPU core, as well as for the BPU, DWT, ITM, ETM and CTI.

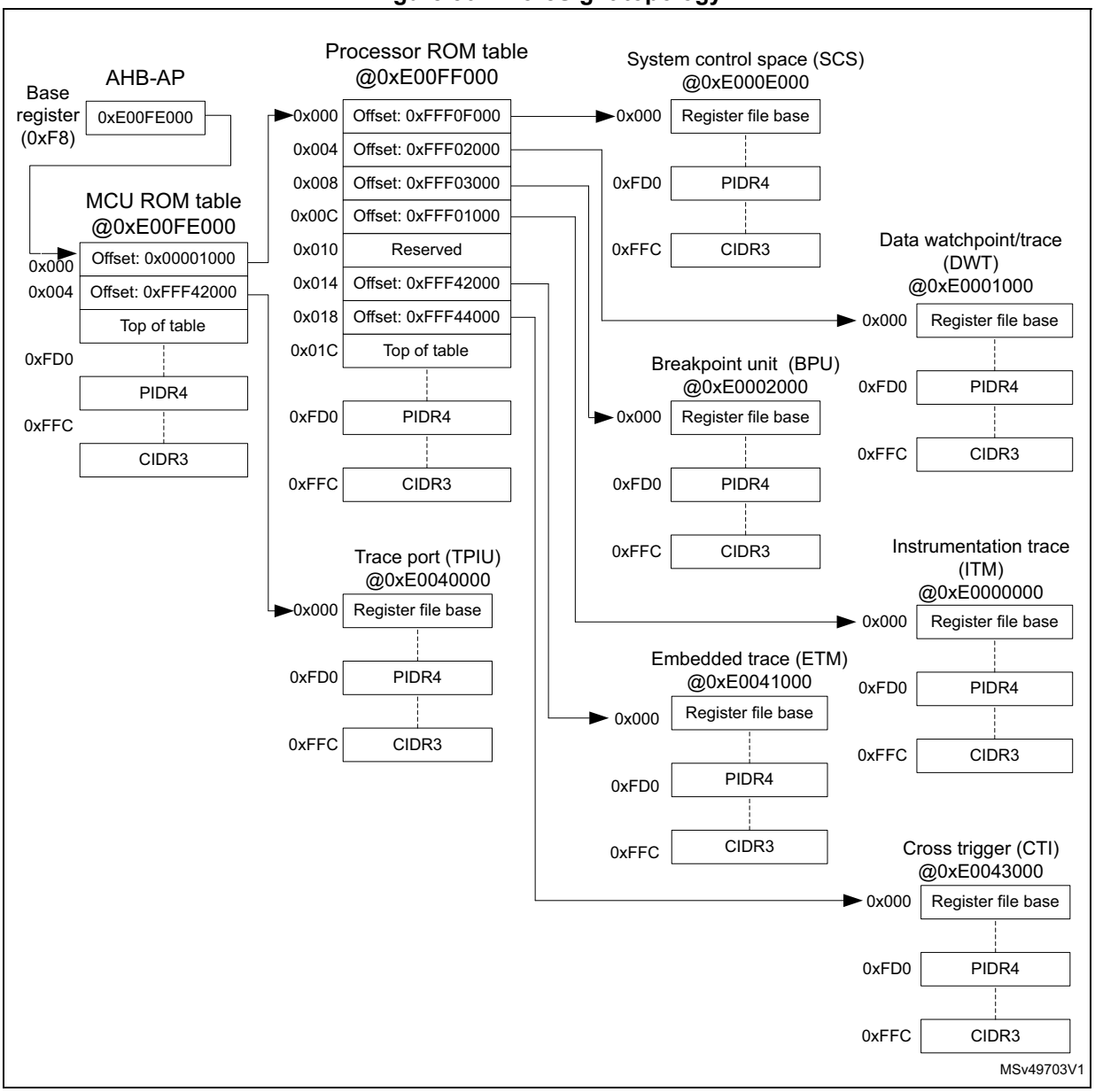
The processor ROM table (see *Table 442*) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F F000 to 0xE00F FFFC.

Table 442. Processor ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F F000	SCS	0xE000 E000	0xFFF0 F000	4	0xFFF0 F003
0xE00F F004	DWT	0xE000 1000	0xFFF0 2000	4	0xFFF0 2003
0xE00F F008	BPU	0xE000 2000	0xFFF0 3000	4	0xFFF0 3003
0xE00F F00C	ITM	0xE000 0000	0xFFF0 1000	4	0xFFF0 1003
0xE00F F010	Reserved	-	-	-	0xFFF4 1002
0xE00F F014	ETM	0xE004 1000	0xFFF4 2000	4	0xFFF4 2003
0xE00F F018	CTI	0xE004 2000	0xFFF4 3000	4	0xFFF4 3003
0xE00F F01C	Reserved	-	-	-	0xFFF4 4002
0xE00F F020	Top of table	-	-	-	0x0000 0000
0xE00F F024 to 0xE00F FFC8	Reserved	-	-	-	0x0000 0000
0xE00F FFCC to 0xE00F FFFC	ROM table registers	-	-	-	See *Table 444*

The topology for the CoreSight components in the Cortex® -M33 is shown in *Figure 564*.

Figure 564. CoreSight topology



52.4.1 MCU ROM table registers

MCU ROM memory type register (MCUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory
0x1: system memory present on this bus

MCU ROM CoreSight peripheral identity register 4 (MCUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

MCU ROM CoreSight peripheral identity register 0 (MCUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0072

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

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

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

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x72: STM32L552xx and STM32L562xx

MCU ROM CoreSight peripheral identity register 1(MCUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: STM32L552xx and STM32L562xx

MCU ROM CoreSight peripheral identity register 2 (MCUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

MCU ROM CoreSight peripheral identity register 3 (MCUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

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

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

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: No customer modifications

MCU ROM CoreSight component identity register 0 (MCUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

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

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: Common identification value

MCU ROM CoreSight peripheral identity register 1 (MCUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

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

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

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

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

MCU ROM CoreSight component identity register 2 (MCUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

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

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

MCU ROM CoreSight component identity register 3 (MCUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

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

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

52.4.2 MCU ROM table register map and reset values

Table 443. MCU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	MCUROM_ MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	MCUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to FDC	Reserved	Reserved																															
0xFE0	MCUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	1	1	0	0	1	0
0xFE4	MCUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE8	MCUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	MCUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	MCUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	MCUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	MCUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	MCUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

52.4.3 Processor ROM table registers

CPU ROM memory type register (CPUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory
1: system memory present on this bus

CPU ROM CoreSight peripheral identity register 4 (CPUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC continuation code

CPU ROM CoreSight peripheral identity register 0 (CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C9

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

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

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0xC9: Cortex®-M33

CPU ROM CoreSight peripheral identity register 1 (CPUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: Cortex®-M33

CPU ROM CoreSight peripheral identity register 2 (CPUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CPU ROM CoreSight peripheral identity register 3 (CPUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

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

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

CPU ROM CoreSight component identity register 0 (CPUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

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

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

CPU ROM CoreSight peripheral identity register 1 (CPUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

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

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

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

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

CPU ROM CoreSight component identity register 2 (CPUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

CPU ROM CoreSight component identity register 3 (CPUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

52.4.4 Processor ROM table register map and reset values

Table 444. CPU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	CPUROM_ MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD4 to FDC	Reserved	Reserved																															
0xFD0	CPUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	CPUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	1	0	0	1	0	0	1
0xFE4	CPUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0xFE8	CPUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CPUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CPUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CPUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	CPUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CPUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

52.5 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as one of the following:

- watchpoint
- ETM trigger
- PC sampling trigger
- data address sampling trigger
- data comparator (comparator 1 only)
- clock cycle counter comparator (comparator 0 only)

It also contains counters for:

- clock cycles
- folded instructions
- load Store Unit (LSU) operations
- sleep cycles
- number of cycles per instruction
- interrupt overhead

A DWT comparator compares the value held in its DWT_COMPR register with one of the following:

- a data address
- an instruction address
- a data value
- the cycle count value, for comparator 0 only

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- one or more DWT data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value
- a watchpoint debug event, on either the PC value or the accessed data address
- a CMPMATCH[N] event, that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v8-M Architecture Reference Manual *[4]*.

52.5.1 DWT registers

The DWT registers are located at address range 0xE000 1000 to 0xE000 1FFC.

DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	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
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: number of comparators implemented (read only)
0x4: four comparators

Bit 27 **NOTRCPKT**: trace sampling and exception tracing support (read only)
0: supported

Bit 26 **NOEXTTRIG**: external match signal, CMPMATCH support (read only)
0: supported

Bit 25 **NOCYCCNT**: cycle counter support (read only)
0: supported

Bit 24 **NOPRFCNT**: profiling counter support (read only)
0: supported

Bit 23 **CYCDISS**: cycle counter disabled secure.
Controls whether the cycle counter is disabled in secure mode.
0: no effect
1: disable incrementing of the cycle counter when the processor is in secure state

Bit 22 **CYCEVTENA**: enable for POSTCNT underflow event counter packet generation
0: disabled
1: enabled

Bit 21 **FOLDEVTENA**: enable for folded instruction counter overflow event generation
0: disabled
1: enabled

Bit 20 **LSUEVTENA**: enable for LSU counter overflow event generation
0: disabled
1: enabled

Bit 19 **SLEEPEVTENA**: enable for sleep counter overflow event generation
0: disabled
1: enabled

Bit 18 **EXCEVTENA**: enable for exception overhead counter overflow event generation
0: disabled
1: enabled

Bit 17 **CPIEVTENA**: enable for CPI counter overflow event generation
0: disabled
1: enabled

Bit 16 **EXCTRCENA**: enable for exception trace generation
0: disabled
1: enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: enable for POSTCNT counter to be used as a timer for periodic PC sample packet generation
0: disabled
1: enabled

Bits 11:10 **SYNCTAP[1:0]**: position of the synchronization packet counter tap on the CYCCNT counter
This field determines the synchronization packet rate.
00: disabled, no synchronization packets
01: Tap at CYCCNT[24]
10: Tap at CYCCNT[26]
11: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Selects the position of the POSTCNT tap on the CYCCNT counter.
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: initial value of the POSTCNT counter
Writes to this field are ignored if POSTCNT counter is enabled. CYCEVTENA or PCSAMPLENA bits must be reset prior to writing POSTINIT.

Bits 4:1 **POSTRESET[3:0]**: reload value of the POSTCNT counter

Bit 0 **CYCCNTENA**: enable CYCCNT counter
0: disabled
1: enabled

DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:0 **CYCCNT[31:0]**: processor clock-cycle counter

DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **CPICNT[7:0]**: CPI counter

Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNTR, and counts any instruction fetch stalls.

DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **EXCCNT[7:0]**: exception overhead cycle counter

Counts the number of cycles spent in exception processing.

DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **SLEEPCNT[7:0]**: sleep cycle counter

Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **LSUCNT[7:0]**: load store counter

Counts additional cycles required to execute load and store instructions.

DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **FOLDCNT[7:0]**: folded instruction counter
Increments on each instruction that takes 0 cycles.

DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

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

Bits 31:0 **EIASAMPLE[31:0]**: executed instruction address sample value.
Samples the current value of the program counter.

DWT comparator x register (DWT_COMPxR)

Address offset: 0x020 + 0x010 * x, (x = 0 to 3)

Reset value: 0xXXXX XXXX

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

Bits 31:0 **COMP[31:0]**: reference value for comparison

DWT function register 0 (DWT_FUNCTR0)

Address offset: 0x028

Reset value: 0x5800 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for match for comparator 0.
0b01011: Cycle Counter, Instruction Address, Data Address and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 0.
For possible values of this field, refer to *[4]*.

DWT function register 1 (DWT_FUNCTR1)

Address offset: 0x038

Reset value: 0xD000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for match for comparator 1.
0b11010: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 1.
For possible values of this field, refer to *[4]*.

DWT function register 2 (DWT_FUNCTR2)

Address offset: 0x048

Reset value: 0x5000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for MATCH for comparator 2
0b01010: Instruction Address, Data Address, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size:
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: Generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 2.
For possible values of this field, refer to *[4]*

DWT function register 3 (DWT_FUNCTR3)

Address offset: 0x058

Reset value: 0xF000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for MATCH for comparator 2.
0b11110: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, Data value, Linked Data Value, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: Generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 2.
For possible values of this field, refer to *[4]*

DWT device type architecture register (DWT_DEVARCHR)

Address offset: 0xFC8

Reset value: 0x4770 1A02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[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
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DWT_DEVARCH register present
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: DWT architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: DWT architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA02: DWT architecture

DWT device type register (DWT_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **SUB[3:0]**: sub-type
0x0: other

Bits 3:0 **MAJOR[3:0]**: major type
0x0: miscellaneous

DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

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

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

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

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: Cortex®-M33 DWT part number

DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: Cortex®-M33 DWT part number

DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

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

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: No customer modifications

DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: Common identification value

DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

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

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

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

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: debug component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

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

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

52.5.2 DWT register map and reset values

The DWT registers are located at address range 0xE000 1000 to 0xE000 1FFC.

Table 445. DWT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DWT_CTRLR	NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DWT_CYCCNTR	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DWT_CPICNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x00C	DWT_EXCCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x010	DWT_SLPCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	DWT_LSUCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x018	DWT_FOLDCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x01C	DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value																																
0x020	DWT_COMP0R	COMP[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
0x024	Reserved	Reserved																															
0x028	DWT_FUNCTR0	ID[4:0]							MATCHED													DATAVSIZE[1:0]							ACTION[1:0]		MATCH[3:0]		
	Reset value	0	1	0	1	1			0													0	0						0	0	0	0	0	0
0x030	DWT_COMP1R	COMP[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
0x034	Reserved	Reserved																															
0x038	DWT_FUNCTR1	ID[4:0]							MATCHED													DATAVSIZE[1:0]							ACTION[1:0]		MATCH[3:0]		
	Reset value	1	1	0	1	0			0													0	0						0	0	0	0	0	0
0x040	DWT_COMP2R	COMP[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
0x044	Reserved	Reserved																															

Table 445. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	DWT_FUNCTR2	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	0			0													0	0					0	0	0	0	0	0
0x050	DWT_COMP3R	COMP[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
0x054	Reserved	Reserved																															
0x058	DWT_FUNCTR3	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	1	1	1	1	0			0													0	0					0	0	0	0	0	0
0x05C to 0xFC4	Reserved	Reserved																															
0xFC8	DWT_DEVARCHR	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				ARCHVER[3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0
0x0FCC	DWT_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	DWT_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	DWT_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0

Table 445. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DWT_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

52.6 Instrumentation trace macrocell (ITM)

The ITM generates trace information in packets. Three sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The three sources in decreasing order of priority are the following:

- Software trace

  The software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.
- Hardware trace

  The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.
- Local timestamping

  The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

For more information on the ITM and how to use it, refer to *[4]*.

52.6.1 ITM registers

The ITM registers are located at address range 0xE000 0000 to 0xE000 0FFC.

ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

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

Bits 31:0 **STIMULUS[31:0]**: trace output data

When writing, write data is output on the trace bus as a software event packet.

When reading:

- bit 1 is a disable flag:
  - 0: stimulus port and ITM enabled
  - 1: stimulus port and ITM disabled
- bit 0 is a FIFO ready indicator:
  - 0: stimulus port buffer is full (or port is disabled)
  - 1: stimulus port can accept new write data

ITM trace enable register (ITM_TER)

Address offset: 0xE00

Reset value: 0x0000 0000

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

Bits 31:0 **STIMENA[31:0]**: stimulus port enable

Each bit x(0 to 31) enables the stimulus port associated with the ITM_STIMRx register.

0: port disabled

1: port enabled

ITM trace privilege register (ITM_TPR)

Address offset: 0xE40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	PRIVMASK[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]**: disable unprivileged access to ITM stimulus ports
Each bit controls eight stimulus ports.
XXX0: unprivileged access permitted on ports 0 to 7
XXX1: only privileged access permitted on ports 0 to 7
XX0X: unprivileged access permitted on ports 8 to 15
XX1X: only privileged access permitted on ports 8 to 15
X0XX: unprivileged access permitted on ports 16 to 23
X1XX: only privileged access permitted on ports 16 to 23
0XXX: unprivileged access permitted on ports 24 to 31
1XXX: only privileged access permitted on ports 24 to 31

ITM trace control register (ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								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.	Res.	TSPRESCALE[1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
						rw	rw			rw	r	rw	rw	rw	rw

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

Bit 23 **BUSY**: indicates whether the ITM is currently processing events
0: not busy
1: busy

Bits 22:16 **TRACEBUSID[6:0]**: identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this field.
Note: Different identifiers must be used for each trace source in the system.

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

Bits 9:8 **TSPRESCALE[1:0]**: local timestamp prescaler, used with the trace packet reference clock
0x0: no prescaling
0x1: Divide by 4.
0x2: Divide by 16.
0x3: Divide by 64.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STALLENA**: stall enable
0: Drop hardware source packets and generate an overflow if the ITM output is stalled.
1: Stall the processor to guarantee delivery of data trace packets.

Bit 4 **SWOENA**: SWO enable
Enables asynchronous clocking of the timestamp counter (read only).
0: Timestamp counter uses processor clock.

Bit 3 **TXENA**: transmit enable
Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: disabled
1: enabled

Bit 2 **SYNCENA**: synchronization packet transmission enable
The debugger setting this bit must also configure the DWT_CTRLR.SYNCTAP field for the correct synchronization speed.
0: disabled
1: enabled

Bit 1 **TSENA**: local timestamp generation enable
0: disabled
1: enabled

Bit 0 **ITMENA**: ITM enable
0: disabled
1: enabled

ITM device type architecture register (ITM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[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
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: ITM architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: ITM architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA01: ITM architecture

ITM device type register (ITM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **SUB[3:0]**: sub-type
0x4: associated with a bus, stimulus derived from bus activity

Bits 3:0 **MAJOR[3:0]**: major type
0x3: trace source

ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

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

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

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

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: ITM part number

ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: ITM part number

ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

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

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

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

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

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

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

52.6.2 ITM register map and reset values

The ITM registers are located at address range 0xE000 0000 to 0xE000 0FFC.

Table 446. CPU1 ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 to 0x07C	ITM_STIM0-31R	STIMULUS[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x07C to 0xDFC	Reserved	Reserved																															
0xE00	ITM_TER	STIMENA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C to 0xDFC	Reserved	Reserved																															
0xE40	ITM_TPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
	Reset value																													0	0	0	0
0xE44 to 0xE7C	Reserved	Reserved																															
0xE80	ITM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0							0	0			0	0	0	0	0	0
0xE84 to 0xFB8	Reserved	Reserved																															
0xFBC	ITM_DEVARCHR	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				ARCHVER[3:0]				ARCHPART[3:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1
0xFC0 to 0xFC8	Reserved	Reserved																															
0xFCC	ITM_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0xFD0	ITM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1

Table 446. CPU1 ITM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	ITM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ITM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	ITM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ITM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ITM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	ITM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ITM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

52.7 Breakpoint unit (BPU)

The BPU allows the user to set hardware breakpoints. It contains eight comparators that monitor the instruction fetch address. If a match occurs, the instruction comparators can be configured to generate a breakpoint instruction.

For more information on the breakpoint unit and how to use it, refer to *[4]*.

52.7.1 BPU registers

The BPU registers are located at address range 0xE0002000 to 0xE0002FFC.

BPU control register (BPU_CTRLR)

Address offset: 0x000

Reset value: 0x1000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

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

Bits 31:28 **REV[3:0]**: revision number
0x1: BPU version 2

Bits 27:15 Reserved, must be kept at reset value.

Bits 11:8 Reserved, must be kept at reset value.

Bits 14:12, 7:4 **NUM_CODE[6:0]**: number of instruction address comparators supported
0x08: 8 instruction comparators supported

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY**: Write protect key
A write to FPB_CTRLR register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: FPB enable
0: disabled
1: enabled

BPU comparator x register (BPU_COMPxR)

Address offset: 0x008 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPADDR[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
BPADDR[15:1]															BE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:1 **BPADDR[31:1]**: breakpoint address

Bit 0 **BE**: breakpoint enable
0: disabled
1: enabled

BPU device type architecture register (BPU_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A03

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[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
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register present
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: BPU architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: BPU architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA03: BPU architecture

BPU device type register (BPU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **SUB[3:0]**: sub-type
0x0: other

Bits 3:0 **MAJOR[3:0]**: major type
0x0: miscellaneous

BPU CoreSight peripheral identity register 4 (BPU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

BPU CoreSight peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

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

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

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

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: BPU part number

BPU CoreSight peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: BPU part number

BPU CoreSight peripheral identity register 2 (BPU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

BPU CoreSight peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

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

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

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

BPU CoreSight component identity register 0 (BPU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

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

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

BPU CoreSight peripheral identity register 1 (BPU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

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

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

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

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: debug component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

BPU CoreSight component identity register 2 (BPU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

BPU CoreSight component identity register 3 (BPU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

52.7.2 BPU register map and reset values

The BPU registers are located at address range 0xE000 2000 to 0xE000 2FFC.

Table 447. CPU1 BPU 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	BPU_CTRLR	REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE[6:4]			Res.	Res.	Res.	Res.	NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	Reset value	0	0	0	1														0	0	0					1	0	0	0			0	0
0x004	Reserved	Reserved																															
0x008 to 0x024	BPU_COMP0R to PBU_COMP7R	BPADDR[31:1]																															ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028 to 0xFB8	Reserved	Reserved																															
0xFBC	BPU_DEVARCHR	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				ARCHVER[3:0]				ARCHPART[3:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1
0xFC0 to 0xFC8	Reserved	Reserved																															
0xFCC	BPU_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	BPU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	BPU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	BPU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	BPU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1

Table 447. CPU1 BPU 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
0xFEC	BPU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	BPU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	BPU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	BPU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	BPU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

52.8 Embedded Trace Macrocell™ (ETM)

The ETM is a CoreSight component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex-M33 core to be traced. In the STM32L552xx and STM32L562xx, the ETM is configured for instruction trace only. Data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- number of instructions executed in the same cycle
- changes in program flow
- current processor instruction state
- addresses of memory locations accessed by load and store instructions
- type, direction and size of a transfer
- Condition code information
- exception information
- wait for interrupt state information

For more information, refer to the Arm® CoreSight™ ETM-M33 Technical Reference Manual *[6]*.

52.8.1 ETM registers

The ETM registers are located at address range 0xE004 1000 to 0xE004 1FFC.

ETM programming control register (ETM_PRGCTLR)

Address offset: 0x004

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: trace unit enable
0: disabled
1: enabled

ETM status register (ETM_STATR)

Address offset: 0x00C

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PMSTABLE**: stability status
Indicates that the ETM-M33 registers are stable and can be read.
0: not stable
1: stable

Bit 0 **IDLE**: trace unit status
Indicates that the trace unit is inactive.
0: not idle
1: idle

ETM configuration register (ETM_CONFIGR)

Address offset: 0x010

Reset value: 0xXXXX 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
Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
			rw		rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RS**: return stack enable
0: disabled
1: enabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:5 **COND[5:0]**: conditional instruction tracing
0x0: conditional instruction tracing disabled
0x1: conditional load instructions traced
0x2: conditional store instructions traced
0x3: conditional load and store instructions traced
0x7: All conditional instructions traced

Bit 4 **CCI**: cycle counting in instruction trace
0: disabled
1: enabled

Bit 3 **BB**: branch broadcast mode
0: disabled
1: enabled

Bits 2:0 Reserved, must be kept at reset value.

ETM event control 0 register (ETM_EVENTCTL0R)

Address offset: 0x020

Reset value: 0xXXXX 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
TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
rw				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: resource type for event1
0: single selected resource
1: boolean combined resource pair

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

Bits 11:8 **SEL1[3:0]**: resource number based on TYPE1
Selects the resource number, based on the value of TYPE1.
When TYPE1 = 0, a single resource from 0-15 defined by SEL1[3:0] is selected.
When TYPE1 = 1, a boolean combined resource pair defined by SEL1[2:0] is selected.

Bit 7 **TYPE0**: resource type for event0
0: single selected resource
1: boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: resource number based on TYPE0
Selects the resource number, based on the value of TYPE0.
When TYPE0 = 0, a single resource from 0-15 defined by SEL0[3:0] is selected.
When TYPE0 = 1, a boolean combined resource pair defined by SEL0[2:0] is selected.

ETM event control 1 register (ETM_EVENTCTL1R)

Address offset: 0x024

Reset value: 0xXXXX 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
Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
			rw	rw										rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: low-power state behavior override
0: normal low-power state behaviour
1: The resources and event trace generation are not affected by entry to a low-power state.

Bit 11 **ATB**: ATB trigger enable
0: disabled
1: enabled

Bits 10:2 Reserved, must be kept at reset value.

Bits 1:0 **INSTEN[1:0]**: instruction event generation
Enables generation of an event element in the instruction stream.
0bX0: Event0 does not cause an event element.
0bX1: Event0 causes an event element when it occurs.
0b0X: Event1 does not cause an event element.
0b1X: Event1 causes an event element when it occurs.

ETM stall control register (ETM_STALLCTLR)

Address offset: 0x02C

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
					rw		rw					rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **INSTPRIORITY**: instruction trace priority
Prioritizes instruction trace if instruction trace buffer space is less than LEVEL.
0: The ETM must not prioritize instruction trace.
1: The ETM can prioritize instruction trace.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ISTALL**: processor stalling
Stalls processor based on instruction trace buffer space.
0: The ETM must not stall the processor.
1: The ETM can stall the processor.

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

Bits 3:0 **LEVEL[3:0]**: Threshold at which stalling becomes active
This field provides four levels. This level can be varied to optimize the level of invasion caused by stalling, balanced against the risk of a FIFO overflow.
0x0: zero invasion, but greater risk of FIFO overflow
...
0xF: maximum invasion but less risk of FIFO overflow

ETM synchronization period register (ETM_SYNCPR)

Address offset: 0x034

Reset value: 0x0000 000A

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

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

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PERIOD[4:0]**: synchronization period

Defines the number of bytes of trace between trace synchronization requests as a total of the number of bytes generated by the instruction stream.
0xA: 1024 bytes

ETM cycle count control register (ETM_CCCTLR)

Address offset: 0x038

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	THRESHOLD[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 **THRESHOLD[11:0]**: instruction trace cycle count threshold

Sets the threshold value for instruction trace cycle counting. The threshold represents the minimum interval between cycle-count trace packets.

ETM trace identification register (ETM_TRACEIDR)

Address offset: 0x040

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace identification to output onto the trace bus

This field must be programmed with a unique value to differentiate it from other trace sources in the system.

Values 0x00 and 0x70-0x7F are reserved.

ETM ViewInst main control register (ETM_VICTLR)

Address offset: 0x080

Reset value: 0xXXXX 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.	EXLEVEL_S[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.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:16 **EXLEVEL_S[3:0]**: exception level in secure state

Controls whether instruction tracing is enabled for the corresponding exception level, in secure state.

0bXXX0: instruction trace not generated in secure state, for exception level 0

0bXXX1: instruction trace generated in secure state, for exception level 0

0b0XXX: instruction trace not generated in secure state, for exception level 3

0b1XXX: instruction trace generated in secure state, for exception level 3

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRCERR**: trace system error exception

0: The system error exception is traced only if the instruction or exception immediately before the system error exception is traced.

1: The system error exception is always traced.

Bit 10 **TRCRESET**: trace reset exception
0: The reset exception is traced only if the instruction or exception immediately before the reset exception is traced.
1: The reset exception is always traced.

Bit 9 **SSSTATUS**: start/stop logic status
0: stopped
1: started

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **EVENT[7:0]**: event selector

ETM counter reload value register 0 (ETM_CNTRLDVR0)

Address offset: 0x140

Reset value: 0xXXXX 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
VALUE[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 **VALUE[15:0]**: counter reload value
This value is loaded in to the counter each time the reload event occurs.

ETM identification register 8 (ETM_IDR8)

Address offset: 0x180

Reset value: 0x0000 0000

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

Bits 31:0 **MAXSPEC[31:0]**: maximum speculation depth
Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.
0x0: The maximum trace speculation depth is zero.

ETM identification register 9 (ETM_IDR9)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP0KEY[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
NUMP0KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP0KEY[31:0]**: number of P0 right-hand keys used
0x0: no P0 right-hand keys used in instruction trace

ETM identification register 10 (ETM_IDR10)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1KEY[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
NUMP1KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1KEY[31:0]**: number of P1 right-hand keys used (including normal and special keys)
0x0: no P1 right-hand keys used in instruction trace

ETM identification register 11 (ETM_IDR11)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1SPC[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
NUMP1SPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1SPC[31:0]**: number of special P1 right-hand keys used
0x0: no special P1 right-hand keys used in any configuration

ETM identification register 12 (ETM_IDR12)

Address offset: 0x190

Reset value: 0x0000 0001

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

Bits 31:0 **NUMCONDKEY[31:0]**: number of conditional instruction right-hand keys used (including normal and special keys)

0x1: one conditional instruction right-hand key implemented

ETM identification register 13 (ETM_IDR13)

Address offset: 0x194

Reset value: 0x0000 0000

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

Bits 31:0 **NUMCONDSPC[31:0]**: number of special conditional instruction right-hand keys used

0x0: no special conditional instruction right-hand keys implemented

ETM implementation specific register 0 (ETM_IMSPECR0)

Address offset: 0x1C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	SUPPORT[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SUPPORT[3:0]**: implementation specific extension support
0x0: no implementation specific extensions are supported

ETM identification register 0 (ETM_IDR0)

Address offset: 0x1E0

Reset value: 0x2800 06E1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1]
		r												r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUPP[0]	Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
r		r	r	r	r	r		r	r	r	r	r	r	r	

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: commit field meaning
Indicates the meaning of the commit field in some packets.
1: commit mode 1

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **TRCEXDATA**: trace data transfers for exceptions
Indicates support for the tracing of data transfers for exceptions and exception returns.
0: not implemented

Bits 16:15 **QSUPP[1:0]**: Q element support
0: not supported

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: conditional results tracing
Indicates how conditional results are traced.
0: The trace unit indicates only if a conditional instruction passes or fails its condition code check

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported
0x1: two events

Bit 9 **RETSTACK**: return stack support
1: two entry return stacks

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: cycle counting support
1: cycle counting implemented

Bit 6 **TRCCOND**: conditional instruction support
1: conditional instruction tracing implemented

Bit 5 **TRCBB**: branch broadcast support
1: branch broadcast tracing implemented

Bits 4:3 **TRCDATA[1:0]**: data tracing support
0x0: data tracing not supported

Bits 2:1 **INSTP0[1:0]**: support for tracing of load and store instructions as P0 elements
0x0: not supported

Bit 0 Reserved, must be kept at reset value.

ETM identification register 1 (ETM_IDR1)

Address offset: 0x1E4

Reset value: 0x4100 F421

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: trace unit designer
0x41: Arm®

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: major trace unit architecture version number
0x4: ETMv4

Bits 7:4 **TRCARCHMIN[3:0]**: minor trace unit architecture version number
0x2: minor revision 2

Bits 3:0 **REVISION[3:0]**: implementation revision number
0x1: implementation revision 1

ETM identification register 2 (ETM_IDR2)

Address offset: 0x1E8

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:1]			
			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
DASIZE[0]	VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: cycle counter size
0x0: 12 bits

Bits 24:20 **DVSIZE[4:0]**: data value size
0x0: data value size not supported

Bits 19:15 **DASIZE[4:0]**: data address size.
0x0: data address size not supported

Bits 14:10 **VMIDSIZE[4:0]**: virtual machine ID size
0x0: virtual machine ID tracing not implemented

Bits 9:5 **CIDSIZE[4:0]**: context ID size
0x0: context ID tracing not implemented

Bits 4:0 **IASIZE[4:0]**: instruction address size
0x4: maximum 32-bit address size

ETM identification register 3 (ETM_IDR3)

Address offset: 0x1EC

Reset value: 0x0F09 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
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.	CCITMIN[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **NOOVERFLOW**: ETM_STALLCTLR.NOOVERFLOW implementation
0: not implemented

Bits 30:28 **NUMPROC[2:0]**: number of processors available for tracing
0x0: one processor

Bit 27 **SYSSTALL**: system support for stall control of the processor
1: system supports stall control

Bit 26 **STALLCTL**: stall control support
1: ETM_STALLCTLR implemented

Bit 25 **SYNCPR**: trace synchronization period support
1: ETM_SYNCPR is read-only for instruction trace only configuration. The trace synchronization period is fixed.

Bit 24 **TRCERR**: ETM_VICTLR.TRCERR implementation
0x1: implemented

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: privilege levels implementation
0x9: privilege levels thread and handler implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CCITMIN[11:0]**: minimum value that can be programmed to TRCCCCTLR.THRESHOLD
Defines the minimum cycle counting threshold.
0x4: minimum of four-instruction trace cycles

ETM identification register 4 (ETM_IDR4)

Address offset: 0x1F0

Reset value: 0x0011 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[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
NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
r	r	r	r				r	r	r	r	r	r	r	r	r

Bits 31:28 **NUMVMIDC[3:0]**: number of virtual machine ID (VMID) comparators
0x0: VMID comparators not implemented

Bits 27:24 **NUMCIDC[3:0]**: number of context ID comparators
0x0: context ID comparators not supported

Bits 23:20 **NUMSSCC[3:0]**: number of single-shot comparator controls
0x1: one single-shot comparator control implemented

Bits 19:16 **NUMRSPAIR[3:0]**: number of resource selection pairs
0x1: two resource selection pairs implemented

Bits 15:12 **NUMPC[3:0]**: number of processor comparator inputs for the DWT
0x4: four processor comparator inputs implemented

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SUPPDAC**: data address comparisons
0: data address comparisons not supported

Bits 7:4 **NUMDVC[3:0]**: number of data value comparators
0x0: no data value comparators implemented

Bits 3:0 **NUMACPAIRS[3:0]**: number of address comparator pairs
0x0: no address comparator pairs implemented

ETM identification register 5 (ETM_IDR5)

Address offset: 0x1F4

Reset value: 0x90C7 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]					
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.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **REDFUNCNTR**: reduced function counter
1: counter 0 implemented as a reduced function counter

Bits 30:28 **NUMCNTR[2:0]**: number of counters
0x1: one counter implemented.

Bits 27:25 **NUMSEQSTATE[2:0]**: number of sequencer states
0x0: no sequencer states implemented.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **LPOVERRIDE**: low-power state override support
1: low-power state override support implemented

Bit 22 **ATBTRIG**: ATB trigger support
1: ATB trigger support implemented

Bits 21:16 **TRACEIDSIZE[5:0]**: number of bits of trace identification
0x7: 7-bit trace identification implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:9 **NUMEXTINSEL[2:0]**: number of external input selectors
0x0: no external input selectors implemented.

Bits 8:0 **NUMEXTIN[8:0]**: number of external inputs
0x004: four external inputs implemented.

ETM resource register 2 (ETM_RSCTLR2)

Address offset: 0x208

Reset value: 0xXXXX 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.	PAIRINV	INV	Res.	GROUP[2:0]		
										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.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PAIRINV**: result of a combined pair of resources inversion
0: not inverted
1: inverted

Bit 20 **INV**: selected resources inversion
0: not inverted
1: inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: group of resources selection
0x0: external input selectors (select 0-3)
0x1: inputs from processor DWT comparators element (select 0-3)
0x2: counter at zero (select 0)
0x3: single-shot comparator (select 0)
others: reserved

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

Bits 7:0 **SELECT[7:0]**: more resources selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM resource register 3 (ETM_RSCTLR3)

Address offset: 0x20C

Reset value: 0xXXXX 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.	INV	Res.	GROUP[2:0]		
											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.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: selected resources inversion
0: not inverted
1: inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: group of resources selection
0x0: external input selectors (select 0-3)
0x1: inputs from processor DWT comparators element (select 0-3)
0x2: counter at zero (select 0)
0x3: single-shot comparator (select 0)
others: reserved

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

Bits 7:0 **SELECT[7:0]**: more resources selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM single-shot comparator control register 0 (ETM_SSCCR0)

Address offset: 0x280

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	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 **RST**: single-shot comparator reset
Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected.
1: reset enabled

Bits 23:0 Reserved, must be kept at reset value.

ETM single-shot comparator status register 0 (ETM_SSCSR0)

Address offset: 0x2A0

Reset value: 0xXXXX XXXX

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
												r	r	r	r

Bit 31 **STATUS**: single-shot comparator status
Indicates whether any of the selected comparators have matched.
0: no match occurred
1: at least one match occurred

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **PC**: processor comparator input sensitivity
1: single-shot comparator sensitive to processor comparator inputs

Bit 2 **DV**: data value comparator support
0: single-shot data value comparisons not supported

Bit 1 **DA**: data address comparator support
0: single-shot data address comparisons not supported

Bit 0 **INST**: instruction address comparator support
0: single-shot instruction address comparisons not supported

ETM single-shot processor comparator input control register 0 (ETM_SSPCICR0)

Address offset: 0x2C0

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PC[3:0]**: processor comparator inputs selection for single-shot control
0xXXX0: processor comparator input 0 not selected
0xXXX1: processor comparator input 0 selected
0xXX0X: Processor comparator input 1 not selected
0xXX1X: processor comparator input 1 selected
0xX0XX: processor comparator input 2 not selected
0xX1XX: processor comparator input 2 selected
0x0XXX: processor comparator input 3 not selected
0x1XXX: processor comparator input 3 selected

ETM power-down control register (ETM_PDCR)

Address offset: 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU**: power-up request
0: power-up not requested
1: power-up requested

Bits 2:0 Reserved, must be kept at reset value.

ETM power-down status register (ETM_PDSR)

Address offset: 0x314

Reset value: 0x0000 0003

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: sticky power-down state
0: Trace register power has not been removed since the ETM_PDSR was last read.
1: Trace register power has been removed since the ETM_PDSR was last read.

Bit 0 **POWER**: ETM power-up status
1: ETM powered up

ETM claim tag set register (ETM_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

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

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: claim tag bits setting
Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

ETM claim tag clear register (ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

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

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: claim tag bits reset
Write:
0000: no effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

ETM authentication status register (ETM_AUTHSTATR)

Address offset: 0xFB8

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

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

Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
0x2: secure non-invasive debug disabled
0x3: secure non-invasive debug enabled

Bits 5:4 **SID[1:0]**: security level for secure invasive debug
0x0: not implemented

Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
0x2: non-secure non-invasive debug disabled
0x3: non-secure non-invasive debug enabled

Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
0x0: not implemented

ETM device type architecture register (ETM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4772 4A13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[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
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x2: ETM architecture v4.2

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x4: ETM architecture v4.2

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA13: ETM architecture

ETM CoreSight device type register (ETM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0013

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

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

Bits 7:4 **SUBTYPE[3:0]**: device sub-type identifier
0x1: processor trace

Bits 3:0 **MAJORTYPE[3:0]**: device main type identifier
0x3: trace source

ETM CoreSight peripheral identity register 4 (ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: ® JEDEC code

ETM CoreSight peripheral identity register 0 (ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

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

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

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

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: ETM part number

ETM CoreSight peripheral identity register 1 (ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: ETM part number

ETM CoreSight peripheral identity register 2 (ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

ETM CoreSight peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

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

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

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

ETM CoreSight component identity register 0 (ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

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

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

ETM CoreSight peripheral identity register 1 (ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

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

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

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: trace generator component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

ETM CoreSight component identity register 2 (ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

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

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

ETM CoreSight component identity register 3 (ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

52.8.2 ETM register map and reset values

The ETM registers are accessed by the debugger at address range 0xE0041000 to 0xE0041FFC.

Table 448. ETM 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
0x004	ETM_PRGCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x008	Reserved	Reserved																															
0x00C	ETM_STATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABL	IDLE
	Reset value																															X	X
0x010	ETM_CONFIGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
	Reset value																				X		X	X	X	X	X	X	X	X			
0x014 to 0x01C	Reserved	Reserved																															
0x020	ETM_ EVENTCTL0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
	Reset value																	X				X	X	X	X	X				X	X	X	X
0x024	ETM_ EVENTCTL1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
	Reset value																				X	X										X	X
0x028	Reserved	Reserved																															
0x02C	ETM_STALLCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
	Reset value																						X		X					X	X	X	X
0x030	Reserved	Reserved																															
0x034	ETM_SYNCPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
	Reset value																												0	1	0	1	0
0x038	ETM_CCCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
	Reset value																					X	X	X	X	X	X	X	X	X	X	X	X
0x03C	Reserved	Reserved																															
0x040	ETM_TRACEIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
	Reset value																										X	X	X	X	X	X	X
0x044 to 0x07C	Reserved	Reserved																															

Table 448. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	ETM_VICTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]				Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
	Reset value													X	X	X	X					X	X	X		X	X	X	X	X	X	X	X
0x084 to 0x13C	Reserved	Reserved																															
0x140	ETM_CNTRLDVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x144 to 0x17C	Reserved	Reserved																															
0x180	ETM_IDR8	MAXSPEC[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
0x184	ETM_IDR9	NUMP0KEY[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
0x188	ETM_IDR10	NUMP1KEY[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
0x18C	ETM_IDR11	NUMP1SPC[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
0x190	ETM_IDR12	NUMCONDKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x194	ETM_IDR13	NUMCONDSPC[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
0x198 to 0x1BC	Reserved	Reserved																															
0x1C0	ETM_IMSPECR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
	Reset value																													0	0	0	0
0x1C4 to 0x1DC	Reserved	Reserved																															
0x1E0	ETM_IDR0	Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1:0]		Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
	Reset value			1												0	0	0		0	0	0	1	1		1	1	1	0	0	0	0	
0x1E4	ETM_IDR1	DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
	Reset value	0	1	0	0	0	0	0	1													0	1	0	0	0	0	1	0	0	0	0	1

Table 448. ETM 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
0x1E8	ETM_IDR2	Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:0]					VMIDSIZE[4:0]			CIDSIZE[4:0]						IASIZE[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	1	0	0
0x1EC	ETM_IDR3	NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]				Res.	Res.	Res.	Res.	CCITMIN[11:0]											
	Reset value	0	0	0	0	1	1	1	1					1	0	0	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F0	ETM_IDR4	NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]				NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC[3:0]			NUMACPAIRS[3:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0				0	0	0	0	0	0	0	0	0
0x1F4	ETM_IDR5	REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]						Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
	Reset value	1	0	0	1	0	0	0		1	1	0	0	0	1	1	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F8 to 0x204	Reserved	Reserved																															
0x208	ETM_RSCTLR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	Res.	GROUP[2:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value											X	X		X	X	X									X	X	X	X	X	X	X	X
0x20C	ETM_RSCTLR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	Res.	GROUP[2:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value												X		X	X	X									X	X	X	X	X	X	X	X
0x210 to 0x27C	Reserved	Reserved																															
0x280	ETM_SSCCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	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								X																								
0x284 to 0x29C	Reserved	Reserved																															
0x2A0	ETM_SSCSR0	STATU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
	Reset value	X																												X	X	X	X
0x2A4 to 0x2BC	Reserved	Reserved																															
0x2C0	ETM_SSPCICR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
	Reset value																													X	X	X	X
0x2C4 to 0x30C	Reserved	Reserved																															

Table 448. ETM 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
0x310	ETM_PDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU			
	Reset value																													0			
0x314	ETM_PDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
	Reset value																															1	1
0x318 to 0xF9C	Reserved	Reserved																															
0xFA0	ETM_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	ETM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFB8	ETM_AUTHSTATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									X	X	X	X	X	X	X	X
0xFBC	ETM_DEVARCHR	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				ARCHVER[3:0]				ARCHPART[3:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	1
0xFC0to 0xFC8	Reserved	Reserved																															
0xFCC	ETM_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYP[3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0xFD0	ETM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4to 0xFDC	Reserved	Reserved																															
0xFE0	ETM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1

Table 448. ETM register map and reset values (continued)

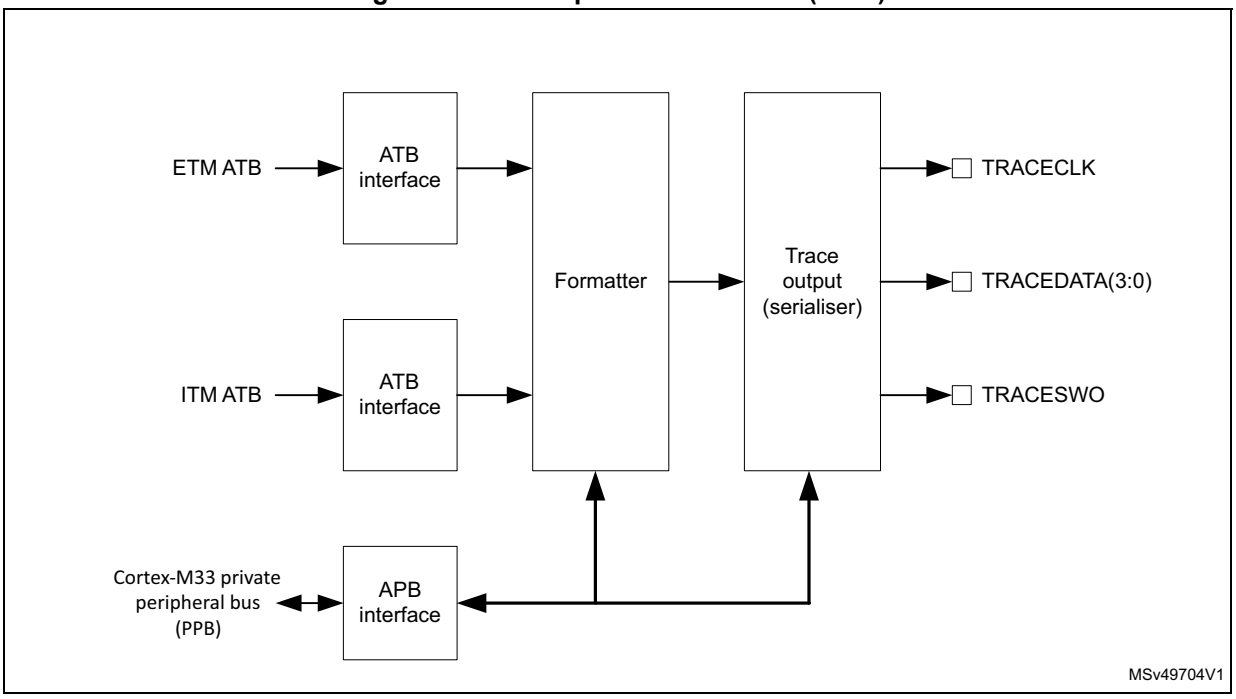
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	ETM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ETM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	ETM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ETM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

52.9 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in *Figure 565*, the TPIU has two ATB slave ports for incoming trace data from the ETM and ITM respectively. The trace port is a synchronous parallel port, comprising a clock output, TRACECLK, and four data outputs, TRACEDATA(3:0). The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up IOs for other purposes, at the expenses of bandwidth restriction of the trace port, and hence of the quantity of trace information that can be output in real time.

Figure 565. Trace port interface unit (TPIU)



Trace data can also be output on the serial-wire output, TRACESWO.

For more information on the trace port interface in the Cortex®-M33, refer to the Arm® Cortex®-M33 Technical Reference Manual *[5]*.

52.9.1 TPIU registers

TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

Reset value: 0x0000 000F

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

Bits 31:0 **PORTSIZE[31:0]**: trace port sizes, from 1 to 32 pins
Bit n-1 when set, indicates that port size n is supported.
0x0000 000F: port sizes 1 to 4 supported

TPIU current port size register (TPIU_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

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

Bits 31:0 **PORTSIZE[31:0]**: current trace port size

Bit n-1 when set, indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result.

This register must only be modified when the formatter is stopped.

TPIU asynchronous clock prescaler register (TPIU_ACPR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: baud rate for the asynchronous output, TRACESWO

The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

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

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TXMODE[1:0]**: protocol used for trace output
0x0: parallel trace port mode
0x1: asynchronous SWO using Manchester encoding
0x2: asynchronous SWO using NRZ encoding
0x3: reserved

TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: formatter stop
Indicates whether formatter can be stopped or not.
1: The formatter cannot be stopped.

Bit 2 **TCPRESENT**: TRACECTL output pin availability
Indicates whether the optional TRACECTL output pin is available for use.
0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: formatter stop
The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.
0: The formatter has not stopped.

Bit 0 **FLINPROG**: flush in progress
Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.
0: no flush in progress
1: flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0102

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
							r		rw					rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **TRIGIN**: trigger on trigger in

1: Indicates a trigger in the trace stream when the TRIGIN input is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: flush on manual

0: flush completed

1: Generates a flush.

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** continuous formatting enable

Setting this bit to zero in SWO mode bypasses the formatter and only ITM/DWT trace is output, ETM trace is discarded.

0: continuous formatting disabled

1: continuous formatting enabled

Bit 0 Reserved, must be kept at reset value.

TPIU periodic synchronization counter register (TPIU_PSCR)

Address offset: 0x308

Reset value: 0x0000 0000

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

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

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PSCOUNT[12:0]**: formatter frames counter

Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word synchronization frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

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

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: claim tag bits setting

Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

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

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: claim tag bits reset
Write:
0000: no effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

TPIU device configuration register (TPIU_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0000 0CA1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: Serial-wire output, NRZ support
0x1: supported

Bit 10 **SWOMAN**: Serial-wire output, Manchester encoded format, support
0x1: supported

Bit 9 **TCLKDATA**: trace clock plus data support
0x0: supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relationship (synchronous or asynchronous)
0x1: asynchronous

Bits 4:0 **MAXNUM[4:0]**: number/type of ATB input port multiplexing
0x1: two input ports

TPIU device type identifier register (TPIU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0011

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

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

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

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x1: trace port component

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x1: trace sink component

TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

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

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

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

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: TPIU part number

TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: TPIU part number

TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

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

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

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

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

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

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

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

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

52.9.2 TPIU register map and reset values

Table 449. CPU1 TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TPIU_SSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0x004	TPIU_CSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008	Reserved	Reserved																															
0x010	TPIU_ACPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x014 to 0x0EC	Reserved	Reserved																															
0x0F0	TPIU_SPPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE[1:0]	
	Reset value																														0	1	
0x0F4 to 0x2FC	Reserved	Reserved																															
0x300	TPIU_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTO	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304	TPIU_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
	Reset value																								1		0					1	
0x308	TPIU_PSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
030C to 0xF9C	Reserved	Reserved																															
0xFA0	TPIU_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0FA8 to 0xFC4	Reserved	Reserved																															
0xFC8	TPIU_DEVIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWONRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MUXNUM[4:0]				
	Reset value																					1	1	0	0	1	0	1	0	0	0	0	1

Table 449. CPU1 TPIU register map and reset values (continued)

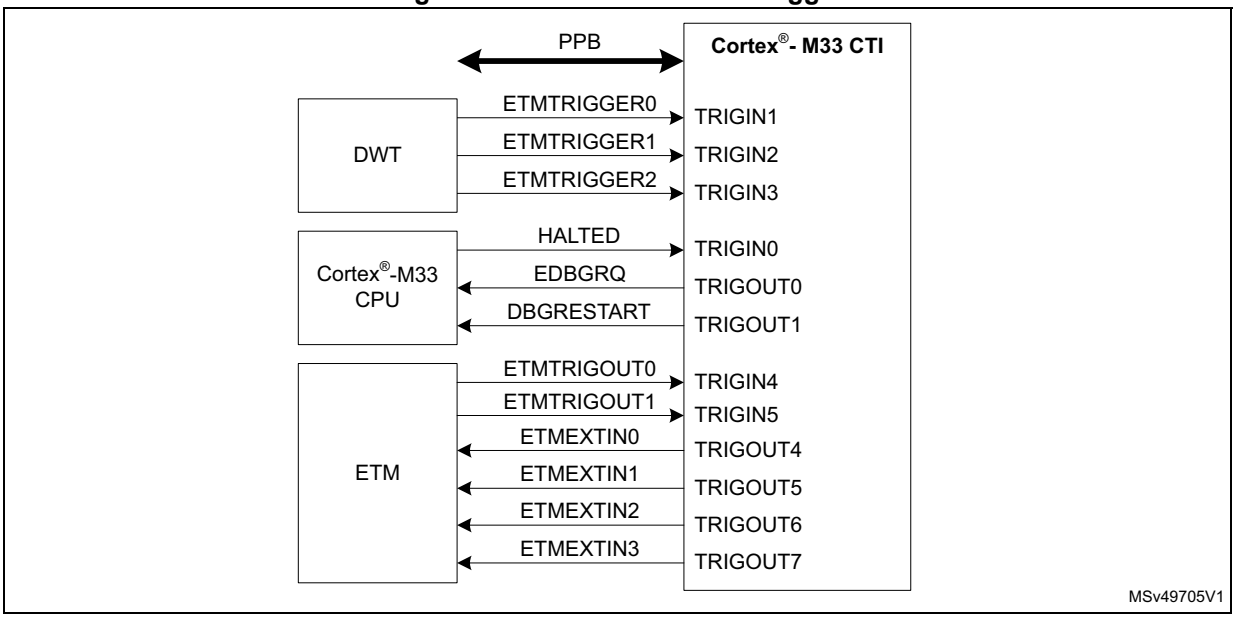
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	TPIU_DEVTYPER	Res.																								SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	TPIU_PIDR4	Res.																								4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	TPIU_PIDR0	Res.																								PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	TPIU_PIDR1	Res.																								JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	TPIU_PIDR2	Res.																								REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	TPIU_PIDR3	Res.																								REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.																								PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TPIU_CIDR1	Res.																								CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.																								PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.																								PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

52.10 Cross-trigger interface (CTI)

The CTI allows cross triggering between the processor and the ETM (see *Figure 566*).

Figure 566. Embedded cross trigger



The CTI enables events from various sources to trigger debug and/or trace activity. For example, a watchpoint reached in the processor can start or stop code trace, or a trace comparator can halt the processor.

The trigger input and output signals for the CTI are listed in *Table 450* and *Table 451*.

Table 450. CTI inputs

Number	Source signal	Source component	Comments
0	HALTED	CPU	Processor halted - CPU is in debug mode
1	ETMTRIGGER0	DWT	DWT comparator output 0
2	ETMTRIGGER1	DWT	DWT comparator output 1
3	ETMTRIGGER2	DWT	DWT comparator output 2
4	ETMTRIGOUT0	ETM	ETM event output 0
5	ETMTRIGOUT1	ETM	ETM event output 1
6	-	-	Not used
7	-	-	Not used

Table 451. CTI outputs

Number	Source signal	Destination component	Comments
0	EDBGRQ	CPU	CPU halt request - Puts CPU in debug mode
1	DBGRESTART	CPU	CPU restart request - CPU exits debug mode

Table 451. CTI outputs (continued)

Number	Source signal	Destination component	Comments
2	ETMEXTIN0	ETM	ETM event input 0
3	ETMEXTIN1	ETM	ETM event input 1
4	ETMEXTIN2	ETM	ETM event input 2
5	ETMEXTIN3	ETM	ETM event input 3
6	-	-	Not used
7	-	-	Not used

For more information on the cross-trigger interface CoreSight component, refer to the Arm® CoreSight SoC-400 Technical Reference Manual *[2]*.

52.10.1 CTI registers

The register file base address for the CTI is 0xE004 2000.

CTI control register (CTI_CONTROLR)

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.	Res.	Res.	Res.	Res.	Res.	GLBEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: global CTI enable
0: disabled
1: enabled

CTI trigger acknowledge register (CTI_INTACKR)

Address offset: 0x010

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								w	w	w	w	w	w	w	w

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

Bits 7:0 **INTACK[7:0]**: trigger acknowledge
There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

CTI application trigger set register (CTI_APPSETR)

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.	Res.	Res.	Res.	APPSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]**: channel event setting
Read:
XXX0: channel 0 event inactive
XXX0: channel 0 event active
XX0X: channel 1 event inactive
XX1X: channel 1 event active
X0XX: channel 2 event inactive
X1XX: channel 2 event active
0XXX: channel 3 event inactive
1XXX: channel 3 event active
Write:
XXX0: no effect
XXX0: Sets event on channel 0.
XX0X: no effect
XX1X: Sets event on channel 1.
X0XX: no effect
X1XX: Sets event on channel 2.
0XXX: no effect
1XXX: Sets event on channel 3.

CTI application trigger clear register (CTI_APPCLEAR)

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.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: channel event clear
0000: no effect
XXX1: Clears event on channel 0.
XX1X: Clears event on channel 1.
X1XX: Clears event on channel 2.
1XXX: Clears event on channel 3.

CTI application pulse register (CTI_APPPULSER)

Address offset: 0x01C

Reset value: 0xXXXX 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]**: pulse channel event
This register clears itself immediately.
0000: no effect
XXX1: Generates pulse on channel 0.
XX1X: Generates pulse on channel 1.
X1XX: Generates pulse on channel 2.
1XXX: Generates pulse on channel 3.

CTI trigger input x enable register (CTI_INENxR)

Address offset: 0x020 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	TRIGINEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]**: trigger input event enable

Enables or disables a cross trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

0000: Trigger does not generate events on channels.

XXX1: Trigger x generates events on channel 0.

XX1X: Trigger x generates events on channel 1.

X1XX: Trigger x generates events on channel 2.

1XXX: Trigger x generates events on channel 3.

CTI trigger output x enable register (CTI_OUTENxR)

Address offset: 0x0A0 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: trigger output event enable

For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

0000: Channel events do not generate triggers on trigger outputs.

XXX1: Channel 0 events generate triggers on trigger output x.

XX1X: Channel 1 events generate triggers on trigger output x.

X1XX: Channel 2 events generate triggers on trigger output x.

1XXX: Channel 3 events generate triggers on trigger output x.

CTI trigger input status register (CTI_TRGISTSR)

Address offset: 0x130

Reset value: 0x0000 0000

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

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

Bits 7:0 **TRIGINSTATUS[7:0]**: trigger input status

There is one bit of the register for each CTITRIGINx input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI trigger output status register (CTI_TRGOSTSR)

Address offset: 0x134

Reset value: 0x0000 0000

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

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

Bits 7:0 **TRIGOUTSTATUS[7:0]**: trigger output status

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel input status register (CTI_CHINSTSR)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]**: channel input status

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel output status register (CTI_CHOUTSTSR)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]**: channel output status

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATER)

Address offset: 0x140

Reset value: 0x0000 000F

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

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]**: channel output enable

For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.

0000: Channels events do not propagate.

XXX1: Channel 0 events propagate.

XX1X: Channel 1 events propagate.

X1XX: Channel 2 events propagate.

1XXX: Channel 3 events propagate.

CTI device configuration register (CTI_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r

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

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

Bits 19:16 **NUMCH[3:0]**: number of ECT channels available

0x4: four channels

Bits 15:8 **NUMTRIG[7:0]**: number of ECT triggers available

0x8: height trigger inputs and height trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: number of trigger input/output multiplexers

0x0: none

CTI device type identifier register (CTI_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0014

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

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

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x1: cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0006

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

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

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: CTI part number

CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: CTI part number

CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

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

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

CTI CoreSight peripheral identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

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

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

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

52.10.2 CTI register map and reset values

Table 452. CTI 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	CTI_CONTROLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value																																0
0x004 to 0x00C	Reserved	Reserved																															
0x010	CTI_INTACKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	CTI_APPSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value																													0	0	0	0
0x018	CTI_APPCLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
	Reset value																													0	0	0	0
0x01C	CTI_APPPULSER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
	Reset value																													X	X	X	X
0x020 to 0x03C	CTI_INEN0R to CTI_INEN7R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0
0x040 to 0x09C	Reserved	Reserved																															
0x0A0 to 0x0BC	CTI_OUTEN0R to CTI_OUTEN7R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0
0x0C0 to 0x12C	Reserved	Reserved																															
0x130	CTI_TRGISTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x134	CTI_TRGOSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 452. CTI 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
0x138	CTI_CHINSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHISTATUS[3:0]			
	Reset value																													0	0	0	0
0x13C	CTI_CHOUTSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOSTATUS[3:0]			
	Reset value																													0	0	0	0
0x140	CTI_GATER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
	Reset value																													1	1	1	1
0x144 to 0xFC4	Reserved	Reserved																															
0xFC8	CTI_DEVIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]				NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
	Reset value													0	1	0	0	0	0	0	0	1	0	0	0				0	0	0	0	0
0xFCC	CTI_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFD0	CTI_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFBC	Reserved	Reserved																															
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	1	1	0
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0

Table 452. CTI 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
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

52.11 Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the software) to:

- maintain the clock and power to the processor cores when in low power modes (Sleep, Stop or Standby)
- maintain the clock and power to the system debug and trace components when in low power modes
- stop the clock to certain peripherals (SMBUS timeout, watchdogs, timers, RTC) when either processor core is stopped in debug mode

Note: The DBGMCU is not a standard CoreSight component, consequently it does not appear in the MCU ROM table.

52.11.1 DBGMCU registers

The DBGMCU registers are not reset by a system reset, only by a power-on reset. They are accessible to the debugger via the AHB access port at base address 0xE004 4000.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x00

Reset value: 0x2001 6472

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
0x1000: revision A
0x2000: revision B
0x2001: revision Z

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: device identification
0x472: STM32L552xx and STM32L562xx

DBGMCU configuration register (DBGMCU_CR)

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.	TRACE_MODE[1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
								rw	rw	rw	rw		rw	rw	

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

Bits 7:6 **TRACE_MODE[1:0]**: trace pin assignment
0x0: trace pins assigned for asynchronous mode (TRACESWO)
0x1: trace pins assigned for synchronous mode with a port width of 1 (TRACECK, TRACED0)
0x2: trace pins assigned for synchronous mode with a port width of 2 ((TRACECK, TRACED0-1)
0x3: trace pins assigned for synchronous mode with a port width of 2 ((TRACECK, TRACED0-3)

Bit 5 **TRACE_EN**: trace port and clock enable.
This bit enables the trace port clock, TRACECK.
0: disabled
1: enabled

Bit 4 **TRACE_IOEN**: trace pin enable
0: disabled - trace pins not assigned
1: enabled - trace pins assigned according to the value of TRACE_MODE field

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Allows debug in Standby mode
0: normal operation.
All clocks are disabled and the core powered down automatically in Standby mode.
1: automatic clock stop/power down disabled
All active clocks and oscillators continue to run during Standby mode, and the core supply is maintained, allowing full debug capability. On exit from Standby mode, a system reset is performed.

Bit 1 **DBG_STOP**: Allows debug in Stop mode
0: normal operation.
All clocks are disabled automatically in Stop mode.
1: automatic clock stop disabled
All active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB1 freeze register 1 (DBGMCU_APB1FZR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
rw								rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res.	Res.	Res.	Res.	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 stop in debug
0: normal operation. LPTIM1 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM1 is frozen while CPU is in debug mode.

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **DBG_I2C3_STOP**: I2C3 SMBUS timeout stop in debug
0: normal operation. I2C3 SMBUS timeout continues to operate while CPU is in debug mode.
1: stop in debug. I2C3 SMBUS timeout is frozen while CPU is in debug mode.

Bit 22 **DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug
0: normal operation. I2C2 SMBUS timeout continues to operate while CPU is in debug mode.
1: stop in debug. I2C2 SMBUS timeout is frozen while CPU is in debug mode.

Bit 21 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in debug
0: normal operation. I2C1 SMBUS timeout continues to operate while CPU is in debug mode.
1: stop in debug. I2C1 SMBUS timeout is frozen while CPU is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP**: IWDG stop in debug
0: normal operation. IWDG continues to operate while CPU is in debug mode.
1: stop in debug. IWDG is frozen while CPU is in debug mode.

Bit 11 **DBG_WWDG_STOP**: WWDG stop in debug
0: normal operation. WWDG continues to operate while CPU is in debug mode.
1: stop in debug. WWDG is frozen while CPU is in debug mode.

Bit 10 **DBG_RTC_STOP**: RTC stop in debug
0: normal operation. RTC continues to operate while CPU is in debug mode.
1: stop in debug. RTC is frozen while CPU is in debug mode.

Bits 9:6 Reserved, must be kept at reset value.

Bit 5 **DBG_TIM7_STOP**: TIM7 stop in debug
0: normal operation. TIM7 continues to operate while CPU is in debug mode.
1: stop in debug. TIM7 is frozen while CPU is in debug mode.

Bit 4 **DBG_TIM6_STOP**: TIM6 stop in debug
0: normal operation. TIM6 continues to operate while CPU is in debug mode.
1: stop in debug. TIM6 is frozen while CPU is in debug mode.

Bit 3 **DBG_TIM5_STOP**: TIM5 stop in debug
0: normal operation. TIM5 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM5 is frozen while CPU is in debug mode.

Bit 2 **DBG_TIM4_STOP**: TIM4 stop in debug
0: normal operation. TIM4 continues to operate while CPU is in debug mode.
1: stop in debug. TIM4 is frozen while CPU is in debug mode.

Bit 1 **DBG_TIM3_STOP**: TIM3 stop in debug
0: normal operation. TIM3 continues to operate while CPU is in debug mode.
1: stop in debug. TIM3 is frozen while CPU is in debug mode.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in debug
0: normal operation. TIM2 continues to operate while CPU is in debug mode.
1: stop in debug. TIM2 is frozen while CPU is in debug mode.

DBGMCU APB1 freeze register 2 (DGBMCU_APB1FZR2)

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.	DBG_LPTIM3_STOP	DBG_LPTIM2_STOP	Res.	Res.	Res.	DBG_I2C4_STOP	Res.
									rw	rw				rw	

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DBG_LPTIM3_STOP**: LPTIM3 stop in debug
0: normal operation. LPTIM2 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM2 is frozen while CPU is in debug mode.

Bit 5 **DBG_LPTIM2_STOP**: LPTIM2 stop in debug
0: normal operation. LPTIM2 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM2 is frozen while CPU is in debug mode.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **DBG_I2C4_STOP**: I2C4 stop in debug
0: normal operation. I2C4 continues to operate while CPU is in debug mode.
1: stop in debug. I2C4 is frozen while CPU is in debug mode.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

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.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP
													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 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in debug
0: normal operation. TIM17 continues to operate while CPU is in debug mode.
1: stop in debug. TIM17 is frozen while CPU is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in debug
0: normal operation. TIM16 continues to operate while CPU is in debug mode.
1: stop in debug. TIM16 is frozen while CPU is in debug mode.

Bit 16 **DBG_TIM15_STOP**: TIM15 stop in debug
0: normal operation. TIM15 continues to operate while CPU is in debug mode.
1: stop in debug. TIM15 is frozen while CPU is in debug mode.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **DBG_TIM8_STOP**: TIM8 stop in debug
0: normal operation. TIM8 continues to operate while CPU is in debug mode.
1: stop in debug. TIM8 is frozen while CPU is in debug mode.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in debug
0: normal operation. TIM1 continues to operate while CPU is in debug mode.
1: stop in debug. TIM1 is frozen while CPU is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

52.11.2 DBGMCU register map and reset values

Table 453. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DBGMCU_IDCODE	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1					0	1	0	0	0	1	1	1	0	0	1	0
0x04	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_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
	Reset value																									0	0	0	0		0	0	
0x08	DBGMCU_ APB1FZR1	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res.	Res.	Res.		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
0x0C	DBGMCU_ APB1FZR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM3_STOP	DBG_LPTIM2_STOP	Res.	Res.	Res.	DBG_I2C4_STOP	Res.
	Reset value																										0	0				0	
0x10	DBGMCU_ APB2FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	Res.
	Reset value														0	0	0			0		0											

Refer to *Section 2.3* for register boundary addresses.

52.12 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G, 16th March 2015
3. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009
4. DDI 0553A (ID092917) - Arm® v8-M Architecture Reference Manual, Issue A.f, 29 September 2017
5. 100230_0002_00_en - Arm® Cortex®-M33 Processor r0p2 Technical Reference Manual, Issue 0002-00, 10 May 2017
6. 100232_0001_00_en - Arm® CoreSight™ ETM-M33 r0p1 Technical Reference Manual, Issue 0001-00, 3 February 2017

53 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 STM32L552xx and STM32L562xx microcontrollers.

53.1 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: 0x0BFA 0590

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)

53.2 Flash size data register

Base address: 0x0BFA 05E0

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.

53.3 Package data register

Base address: 0x0BFA 0500

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
00010: LQFP100
00011: UFBGA132
00100: LQFP144
00101: WLCSP81
01011: LQFP48 and UFQFPN48
10001: LQFP144 with SMPS Step Down converter
10010: UFBGA132 with SMPS Step Down converter
10011: LQFP100 with SMPS Step Down converter
10100: WLCSP81 with SMPS Step Down converter
10101: LQFP64 with SMPS Step Down converter
10110: LQFP48 and UFQFPN48 with SMPS Step Down converter
11000: LQFP144 with external SMPS support
11001: UFBGA132 with external SMPS support
11010: LQFP100 with external SMPS support
11011: WLCSP81 with external SMPS support
11100: LQFP64 and with external SMPS support
11101: LQFP48 with external SMPS support
Others: reserved

54 Revision history

Table 454. Document revision history

Date	Revision	Changes
08-Oct-2018	1	Initial release.
12-Apr-2019	2	Added: – *Section 6.5.7: Flash registers privileged and unprivileged modes* – *Section 8.3.4: Upper voltage threshold monitoring* – *Section 8.3.5: Temperature threshold monitoring* – *Section 20.3: OCTOSPI implementation* – *Section 42.3.3: TAMP register write protection* – *Section 43.4.2: I2C pins and internal signals* – *Section 43.7.12: I2C hardware configuration register (I2C_HWCFGR)* – *Section 43.7.13: I2C version register (I2C_VERR)* – *Section 43.7.14: I2C identification register (I2C_IPIDR)* – *Section 43.7.15: I2C size identification register (I2C_SIDR)* – *Section 48.8: SDMMC interrupts* Updated: – *Table 4: STM32L552xx and STM32L562xx memory map and peripheral register boundary addresses* – *Table 7: Boot modes when TrustZone is enabled (TZEN=1)* – *Table 8: Boot space versus RDP protection* – Major changes to all sections (including figures, tables and registers) of *Section 5: Global TrustZone® controller (GTZC)* – *Section 6.1: Introduction* – *Section 6.2: FLASH main features* – *Table 30: Flash module - 512 KB dual bank organization (64 bits read width)* – *Table 31: Flash module - 512 KB single bank organization (128 bits read width)* – *Section 6.3.5: Flash program and erase operations* – *Section 6.3.8: Flash errors flags* – *Section 6.4.2: Option bytes programming* – *Section 6.5.1: TrustZone security protection* – *Section 6.5.2: Secure watermark-based area protection* – *Table 35: Secure watermark-based area* – *Table 22: Secure hide protection* – *Section 5.5.4: Secure proprietary code readout protection (PCROP)* – *Section 6.5.4: Secure block-based area (SECBB) protection* – *Section 6.5.6: Flash security attribute state* – *Section 6.7: FLASH memory protection* – *Section 6.7.1: Write protection (WRP)* – Major updates on content and tables of subsections of *Section 6.7.2: Readout protection (RDP)*

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Apr-2019	2 (continued)	Updated: – *Table 41: Flash memory readout protection status (TZEN=0)* – *Table 42: Access status versus protection level and execution modes when TZEN=0* – *Table 43: Flash memory readout protection status (TZEN=1)* – *Table 44: Access status versus protection level and execution modes when TZEN=1* – *Section 6.9.8: Flash status register (FLASH_SECSR)* – *Section 6.9.12: Flash option register (FLASH_OPTR)* – *Section 7.4.2: ICACHE reset and clocks* – *Section 7.4.5: ICACHE enable* – *Section 7.4.7: Address remapping* – *Figure 20: ICACHE remapping address mechanism* – *Table 59: ICACHE register map and reset values* – *Section 8: Power control (PWR)* – *Section 8.2.5: Dynamic voltage scaling management* – *Table 65: Functionalities depending on the working mode* – *Section 8.5: PWR TrustZone security* – *Section 11.6.6: GPIO port output data register (GPIOx_ODR) (x = A to H)* – *Section 11.6.12: GPIO secure configuration register (GPIOx_SECCFGR) (x = A to H)* – *Table 87: GPIO register map and reset values* – *Section 12.2: SYSCFG TrustZone security and privilege* – *Section Table 89.: BOOSTEN and ANASWVDD set/reset* – *Section 14.4.5: DMA channels* – *Section 14.6.3: DMA channel x configuration register (DMA_CCRx)* – *Table 97: DMAMUX instantiation* – *Section 15.4.6: DMAMUX request line multiplexer* – *Section 15.6.4: DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)* – *Section 15.6.5: DMAMUX request generator interrupt status register (DMAMUX_RGSR)* – *Table 104: STM32L552xx and STM32L562xx vector table* – All registers and tables of *Section 18.4: CRC registers* – *Table 114: CRC register map and reset values* – *Section 20.1: Introduction* – *Section 20.4.1: OCTOSPI block diagram* including all figures – *Section 20.4.2: OCTOSPI interface to memory modes* – *Section 20.4.4: OCTOSPI Regular-command protocol signal interface* – *Section 20.4.6: Specific features* – *Section 20.4.7: OCTOSPI operating modes introduction* – *Section 20.4.9: OCTOSPI Automatic status-polling mode*

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Apr-2019	2 (continued)	Updated: – *Section 20.4.11: OCTOSPI configuration introduction* – *Section 20.4.13: OCTOSPI device configuration* – *Section 20.4.14: OCTOSPI Regular-command mode configuration* – *Section 20.4.16: OCTOSPI error management* – *Section 20.4.17: OCTOSPI BUSY and ABORT* – *Section 20.7.1: OCTOSPI control register (OCTOSPI_CR)* – *Section 20.7.2: OCTOSPI device configuration register 1 (OCTOSPI_DCR1)* – *Section 20.7.5: OCTOSPI device configuration register 4 (OCTOSPI_DCR4)* – *Section 21.2: ADC main features* – *Section 21.4.7: Single-ended and differential input channels* – *Section 21.4.9: ADC on-off control (ADEN, ADDIS, ADRDY)* – *Section 21.4.12: Channel-wise programmable sampling time (SMPR1, SMPR2)* – *Section 21.4.16: ADC timing* – *Table 163: TSAR timings depending on resolution* – *Table 167: Analog watchdog 2 and 3 comparison* – *Table 166: Analog watchdog 1 comparison* – *Figure 134: ADCy_AWDx_OUT signal generation (on all injected channels)* – *Section 21.4.31: Dual ADC modes* – *Figure 145: Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode* – *Figure 146: Interleaved mode on 1 channel in single conversion mode: dual ADC mode* – *Section 21.4.34: Monitoring the internal voltage reference* – *Section 22.4.12: Dual DAC channel conversion modes (if dual channels are available)* – *Section 22.7.17: DAC channel1 sample and hold sample time register (DAC_SHSR1)* – *Section 22.7.18: DAC channel2 sample and hold sample time register (DAC_SHSR2)* – *Section 24.6.1: Comparator 1 control and status register (COMP1_CSR)* – *Section 24.6.2: Comparator 2 control and status register (COMP2_CSR)* – *Table 197: DFSDM1 implementation* – Major changes to all sections (including figures, tables and registers) of *Section 28: True random number generator (RNG)* – *Section 29.4.5: AES decryption round key preparation* – *Section 29.4.8: AES basic chaining modes (ECB, CBC)* – *Section 29.4.10: AES Galois/counter mode (GCM)* – *Figure 213: Message construction in GMAC mode* – *Figure 214: GMAC authentication mode*

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Apr-2019	2 (continued)	Updated: – *Figure 215: Message construction in CCM mode* – *Section 29.4.12: AES counter with CBC-MAC (CCM)* – *Section 29.4.14: AES key registers* – *Section 29.4.16: AES DMA interface* – *Section 29.7.2: AES status register (AES_SR)* – *Section 30.7.4: HASH digest registers* – *Section 30.7.7: HASH context swap registers* – Major changes to all sections (including figures, tables and registers) of *Section 32: Public key accelerator (PKA)* – *Section 33.3.8: PWM input mode* – *Section 33.4.2: TIMx control register 2 (TIMx_CR2)(x = 1, 8)* – *Section 33.4.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)* – *Section 33.4.21: TIMx DMA control register (TIMx_DCR)(x = 1, 8)* – *Figure 289: General-purpose timer block diagram* – *Figure 315: Capture/Compare channel 1 main circuit* – *Section 34.3.3: Clock selection* – *Figure 316: Output stage of Capture/Compare channel (channel 1)* – *Section 34.4.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5)* – *Section 34.4.2: TIMx control register 2 (TIMx_CR2)(x = 2 to 5)* – *Section 34.4.5: TIMx status register (TIMx_SR)(x = 2 to 5)* – *Section 34.4.7: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)* – *Section 34.4.8: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)* – *Section 34.4.9: TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 5)* – *Section 34.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)* – *Section 34.4.12: TIMx counter [alternate] (TIMx_CNT)(x = 2 to 5)* – *Section 34.4.13: TIMx counter [alternate] (TIMx_CNT)(x = 2 to 5)* – *Figure 339: TIM15 block diagram* – *Figure 366: Output redirection* – *Section 35.5.2: TIM15 control register 2 (TIM15_CR2)* – *Section 35.5.5: TIM15 status register (TIM15_SR)* – *Section 35.5.7: TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)* – *Section 35.5.8: TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)* – *Section 35.5.9: TIM15 capture/compare enable register (TIM15_CCER)* – *Section 35.5.16: TIM15 break and dead-time register (TIM15_BDTR)* – *Section 35.6.3: TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)*

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Apr-2019	2 (continued)	Updated: – – *Section 35.6.4: TIMx status register (TIMx_SR)(x = 16 to 17)* – *Section 35.6.6: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)* – *Section 35.6.7: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)* – *Section 35.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)* – *Section 35.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)* – All registers on *Section 36.4: TIM6/TIM7 registers* – *Figure 383: Low-power timer block diagram* – *Table 293: Effect of low-power modes on the LPTIM* – *Section 37.7.1: LPTIM interrupt and status register (LPTIM_ISR)* – *Section 37.7.2: LPTIM interrupt clear register (LPTIM_ICR)* – *Section 37.7.3: LPTIM interrupt enable register (LPTIM_IER)* – *Section 37.7.4: LPTIM configuration register (LPTIM_CFGR)* – Major changes to all sections (including figures, tables and registers) of *Section 39: Independent watchdog (IWDG)* – *Table 299: RTC internal input/output signals* – *Section 41.6.2: RTC date register (RTC_DR)* – *Figure 396: TAMP block diagram* – *Table 309: TAMP internal input/output signals* – *Table 310: TAMP interconnection* – *Section 42.3.6: Tamper detection* – *Section 42.6.1: TAMP control register 1 (TAMP_CR1)* – *Section 42.6.11: TAMP interrupt enable register (TAMP_IER)* – *Section 42.6.12: TAMP status register (TAMP_SR)* – *Section 42.6.13: TAMP non-secure masked interrupt status register (TAMP_MISR)* – *Section 42.6.14: TAMP secure masked interrupt status register (TAMP_SMISR)* – *Table 332: Effect of low-power modes on the I2C* – *Table 333: I2C Interrupt requests* – *Section 43.7.2: I2C control register 2 (I2C_CR2)* – *Section 43.7.3: I2C own address 1 register (I2C_OAR1)* – *Table 334: I2C register map and reset values* – *Section 44: Universal synchronous/asynchronous receiver transmitter (USART/UART)* – *Section 44.5.4: USART FIFOs and thresholds* – *Section 44.5.6: USART receiver* – *Figure 449: Reception using DMA* – *Table 359: USART interrupt requests* – Registers and tables of *Section 44.8: USART registers*

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Apr-2019	2 (continued)	Updated: – *Section 45.4.4: LPUART FIFOs and thresholds* – *Figure 463: Reception using DMA* – *Table 366: LPUART interrupt requests* – Registers and tables of *Section 45.7: LPUART registers* – *Section 46.2: SPI main features* – *Section 46.3: SPI implementation* – *Section :* – *Figure 495: SAI_ADR format in TDM, 32-bit slot width* – *Section 47.4.12: SPDIF output* – *Section 48.4.4: SDMMC adapter* – *Table 382: Data path status flags and clear bits* – *Section 48.4.6: SDMMC AHB master interface* – *Section 54.5.8: Hardware flow control* – *Section 48.5.2: CMD12 send timing* – *Section 48.9.10: SDMMC data counter register (SDMMC_DCNTR)* – *Section 48.9.11: SDMMC status register (SDMMC_STAR)* – Major changes to all sections (including figures, tables and registers) of *Section 49: FD controller area network (FDCAN)* – *Table 408: STM32L552xx and STM32L562xx USB implementation* – *Section 50.6.1: Common registers* – *Section 50.6.2: Buffer descriptor table* – *Section Table 418.: USB register map and reset values* – *Section 51.7.4: UCPD control register (UCPD_CR)*

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Jun-2019	3	Added: – *Section 8.2.6: VDD12 domain and external SMPS* – *Table 252: ECC Fp scalar multiplication (Fast Mode)* – *Figure 25: Internal main regulator overview* – *Section 47.3: SAI implementation* – *Table 356: STM32L5 Series SAI features* – *Table 359: External synchronization selection* – *Table 360: MCLK_x activation conditions* – *Table 363: Allowed TDM frame configuration* Updated: – *Section 6.2: FLASH main features* – *Section 6.3.6: Flash main memory erase sequences* – *Section 6.3.8: Flash errors flags* – *Table 33: User option byte organization mapping* – *Section 6.4.2: Option bytes programming* – *Section 6.5.1: TrustZone security protection* – *Table 34: Default secure option bytes after TZEN activation* – *Table 22: Secure hide protection* – *Table 36: Secure, HDP protections summary* – *Section 6.7: FLASH memory protection* – *Section 6.7.1: Write protection (WRP)* – *Table 45: Flash access versus RDP level when TrustZone is active (TZEN=1)* – *Table 46: Flash access versus RDP level when TrustZone is disabled (TZEN=0)* – *Table 47: Flash mass erase versus RDP level when TrustZone is active (TZEN = 1)* – *Section 6.9.7: Flash status register (FLASH_NSSR)* – *Section 6.9.8: Flash status register (FLASH_SECSR)* – *Section 6.9.10: Flash secure control register (FLASH_SECCR)* – *Section 6.9.12: Flash option register (FLASH_OPTR)* – *Section 6.9.17: Flash secure watermak1 register 2 (FLASH_SECWM1R2)* – *Section 6.9.21: Flash secure watermak2 register 2 (FLASH_SECWM2R2)* – *Table 51: Flash interface - register map and reset values* – *Section 8.1: Power supplies and supply domains* – *Section 8.2.2: Embedded SMPS step down converter* – *Section 8.2.3: SMPS step down converter power supply scheme* – *Section 9.8.10: RCC AHB1 peripheral reset register (RCC_AHB1RSTR)* – *Table 80: RCC register map and reset values* – *Table 82: Effect of low-power modes on CRS* – *Section 12.3.11: SYSCFG RSS command register (SYSCFG_RSSCMDR)* – *Table 90: SYSCFG register map and reset values* – *Section 20.4.14: OCTOSPI Regular-command mode configuration* – *Section 20.7.1: OCTOSPI control register (OCTOSPI_CR)* – *Section 20.7.2: OCTOSPI device configuration register 1 (OCTOSPI_DCR1)*

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Jun-2019	3 (continued)	Updated (cont): – *Section 24.6.1: Comparator 1 control and status register (COMP1_CSR)* – *Table 186: COMP1 input plus assignment* – *Table 192: COMP register map and reset values* – *Section 28.2: RNG main features* – *Section 28.3.3: Random number generation* – *Section 28.3.5: RNG operation* – *Section 28.5: RNG processing time* – *Table 28.5: RNG processing time* – *Table 214: RNG configurations* – *Section 28.7.4: RNG health test control register (RNG_HTCR)* – *Section 29.1: Introduction* – *Section 29.4.3: AES cryptographic core* – *Section 29.4.4: AES procedure to perform a cipher operation* – *Section 29.4.8: AES basic chaining modes (ECB, CBC)* – *Section 29.4.12: AES counter with CBC-MAC (CCM)* – *Table 222: AES interrupt requests* – *Section 30.4.5: Message digest computing* – *Section 30.7.1: HASH control register (HASH_CR)* – *Section 30.7.2: HASH data input register (HASH_DIN)* – Major changes to all *Section 31: On-the-fly decryption engine (OTFDEC)* – Major changes to all *Section 32: Public key accelerator (PKA)* – *External clock source mode 2* on *Section 33.3.5: Clock selection* – *Section 33.4.5: TIMx status register (TIMx_SR)(x = 1, 8)* – *Section 33.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)* – Major changes to all *Section 37: Low-power timer (LPTIM)* – *Section 42.2: TAMP main features* – *Table 310: TAMP interconnection* – *Section 44.8.9: USART interrupt and status register [alternate] (USART_ISR)* – *Section 44.8.12: USART receive data register (USART_RDR)* – *Section 44.8.13: USART transmit data register (USART_TDR)* – Major changes to all *Section 47: Serial audio interface (SAI)* – Major changes to all *Section 51: USB Type-C™ / USB Power Delivery interface (UCPD)* – *Section 52.11.1: DBGMCU registers* Deleted: – Sections ECC complete addition, ECC double base ladder and ECC projective to affine from Section 22: Public key accelerator (PKA) – Tables Modular exponentiation (protected mode), ECC complete addition, ECC double base ladder, ECC projective to affine from Section 22: Public key accelerator (PKA) – Section Secure proprietary code readout protection (PCROP) from Section 5: Embedded Flash memory (FLASH)

Table 454. Document revision history (continued)

Date	Revision	Changes
11-Oct-2019	4	Updated: – *Section 2: Memory and bus architecture* introduction – *Figure 2: Memory map based on IDAU mapping.* – *Section 3: Boot configuration* – *Section 5.3.5: TrustZone illegal access controller (TZIC)* – *Section 5.3.6: Power-on/reset state* – *Section 5.3.5: TrustZone illegal access controller (TZIC)* – *Section 5.5.6: GTZC_TZSC external memory x non-secure watermark register 1 (GTZC_TZSC_MPCWMxANSR)* – *Section 5.5.7: GTZC_TZSC external memory x non-secure watermark register 2 (GTZC_TZSC_MPCWMxBNSR)* – *Section 5.6.2: GTZC_MPCBB1 lock register 1(GTZC_MPCBB1_LCKVTR1)* – *Section : Rules for modifying specific option bytes* – *Section : Level 1: readout protection* – *Section : Level 2: no debug* – *Section : Level 1: readout protection* – *Section : Level 2: no debug* – *Section 6.9.15: Flash secure boot address 0 register (FLASH_SECBOOTADD0R)* – *Section 8.2.2: Embedded SMPS step down converter* – *Table 60: SMPS modes summary* – *Section 8.2.3: SMPS step down converter power supply scheme* – *Section 8.2.4: SMPS step down converter versus low-power mode* – *Section 8.2.5: Dynamic voltage scaling management* – *Section 9.3: Clocks* introduction – *Section 9.3.3: MSI clock* – Caution note in *Section 9.8.4: RCC PLL configuration register (RCC_PLLCFGR)* – *Section 10.1: Introduction* – *Section 10.4.3: Frequency error measurement* – *Section 10.4.5: CRS initialization and configuration* – *Section 10.7: CRS registers* – *Section 11.3: GPIO functional description* – *Section 11.4: TrustZone security* – *Section 11.6.3: GPIO port output speed register (GPIOx_OSPEEDR) (x = A to H)* – *Section 17.6.7: EXTI privilege configuration register (EXTI_PRIVCFGR1)* – *Section 17.6.16: EXTI lock register (EXTI_LOCKR)* – *Section 17.6.16: EXTI lock register (EXTI_LOCKR)* – *Section 19.6.4: NOR Flash/PSRAM controller asynchronous transactions* – *Section 19.6.5: Synchronous transactions* – *Figure 91: ADC block diagram* – *Table 157: ADC internal input/output signals* – *Section 21.4.7: Single-ended and differential input channels* – *Section 21.4.11: Channel selection (SQRx, JSQRx)* – *Section 21.4.20: Discontinuous mode (DISCEN, DISCNUM, JDISCEN)* – *Section 21.4.26: Data management* – *Section 21.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Table 454. Document revision history (continued)

Date	Revision	Changes
11-Oct-2019	4 (continued)	– *Section 21.6.6: ADC sample time register 1 (ADC_SMPR1)* – *Section 21.6.7: ADC sample time register 2 (ADC_SMPR2)* – *Section 21.6.11: ADC regular sequence register 1 (ADC_SQR1)* – *Section 21.6.12: ADC regular sequence register 2 (ADC_SQR2)* – *Section 21.6.13: ADC regular sequence register 3 (ADC_SQR3)* – *Section 21.6.14: ADC regular sequence register 4 (ADC_SQR4)* – *Section 21.6.16: ADC injected sequence register (ADC_JSQR)* – *Section 21.6.17: ADC offset y register (ADC_OFRy)* – *Section 21.6.19: ADC analog watchdog 2 configuration register (ADC_AWD2CR)* – *Section 21.6.20: ADC analog watchdog 3 configuration register (ADC_AWD3CR)* – *Section 21.6.21: ADC differential mode selection register (ADC_DIFSEL)* – *Section 22.3: DAC implementation* – *Section 22.4.6: DAC trigger selection* – *Section 22.4.10: DAC channel modes* – *Section 22.4.11: DAC channel buffer calibration* – *Section 22.4.12: Dual DAC channel conversion modes (if dual channels are available)* – *Table 181: DAC interrupts* – *Section 22.7.1: DAC control register (DAC_CR)* – *Section 26.2: DFSDM main features* – *Section 26.4.3: DFSDM reset and clocks* – *Section 26.7: DFSDM channel y registers (y=0..3)* introduction – *Section 26.7.5: DFSDM channel y data input register (DFSDM_CHyDATINR)* – *Section 26.8: DFSDM filter x module registers (x=0..3)* introduction – *Table 207: Spread spectrum deviation versus AHB clock frequency* – *Section 28.2: RNG main features* – *Section 28.3.5: RNG operation* – *Section 28.5: RNG processing time* – *Section 28.7.3: RNG data register (RNG_DR)* – *Figure 209: CTR encryption* – *Section 30: Hash processor (HASH)* – *Section 33.3.3: Repetition counter* – *Table 275: TIMx internal trigger connection* – *Section 35.5: TIM15 registers* introduction – *Section 35.6: TIM16/TIM17 registers* introduction – *Figure 383: Low-power timer block diagram* – *Section 37.4.16: Repetition Counter* – *Section 37.6: LPTIM interrupts* – *Section 37.7.8: LPTIM counter register (LPTIM_CNT)* – *Section 40: System window watchdog (WWDG)* – *Section 42.6.2: TAMP control register 2 (TAMP_CR2)* – *Table 359: USART interrupt requests* – *Table 44.8: USART registers* introduction – *Table 45.7: LPUART registers* introduction – *Table 357: SAI internal input/output signals* – *Figure 494: Start-up sequence*

Table 454. Document revision history (continued)

Date	Revision	Changes
11-Oct-2019	4 (continued)	– *Table 367: SAI interrupt sources* – *Section 47.6: SAI registers* introduction Added – *Section 6.5.5: Forcing boot from a secure memory address* – *Section 10.3: CRS implementation* Deleted: – 3.6.3 GTZC_MPCBBx lock register 2 (GTZC_MPCBBx_LCKVTR2) (x = 1 to 2)
13-Feb-2020	5	Updated cover page. Updated *Section 2: Memory and bus architecture* introduction. Updated *Section 2.1: System architecture* introduction. Updated *Section Figure 1.: System architecture*. Updated *Section 3: Boot configuration*. Added *Section 4: System security*. **FLASH:** Updated *Section 6.3.1: Flash memory organization*. Updated *Table 30: Flash module - 512 KB dual bank organization (64 bits read width)*. Updated *Table 31: Flash module - 512 KB single bank organization (128 bits read width)*. Updated *Section 6.3.5: Flash program and erase operations*. Updated *Section 6.3.6: Flash main memory erase sequences*. Updated *Section 6.3.7: Flash main memory programming sequences*. Updated *Section 6.3.8: Flash errors flags*. Updated *Section 6.4.1: Option bytes description*. Updated *Section 6.4.2: Option bytes programming*. Updated *Section 6.5.1: TrustZone security protection*. Updated *Section 6.5.3: Secure hide protection (HDP)*. Updated *Table 36: Secure, HDP protections summary*. Updated *Section 6.5.4: Secure block-based area (SECBB) protection*. Added *Section 6.6: Secure system memory*. Updated *Section 6.7.2: Readout protection (RDP)*. Updated *Table 41: Flash memory readout protection status (TZEN=0)*. Updated *Table 43: Flash memory readout protection status (TZEN=1)*. Updated *Table 48: Flash system memory, RSS and OTP accesses*. Updated *Section 6.9.7: Flash status register (FLASH_NSSR)*. Updated *Section 6.9.11: Flash ECC register (FLASH_ECCR)*. **PWR:** Updated *Figure 21: STM32L552xx and STM32L562xx power supply overview*. Updated *Figure 22: STM32L552xxxP and STM32L562xxxP power supply overview*. Updated *Table 62: SMPS step down converter versus low-power modes*. Updated *Section 8.2.6: VDD12 domain and external SMPS*. Updated *Section 8.3.4: Upper voltage threshold monitoring*. Updated *Section 8.3.5: Temperature threshold monitoring*. **RCC:** Updated *Section 9.8.1: RCC clock control register (RCC_CR)*. Added *Section 9.8.33: OCTOSPI delay configuration register (RCC_DLYCFGR)*.

Table 454. Document revision history (continued)

Date	Revision	Changes
13-Feb-2020	5 (continued)	**SYSCFG:** Updated *Section 12.3.11: SYSCFG RSS command register (SYSCFG_RSSCMDR)*. **NVIC:** Updated *Section 16.1: NVIC main features*. **OCTOSPI:** Updated entire *Section 20: Octo-SPI interface (OCTOSPI)*. **ADC:** Updated *Section 21.2: ADC main features*. Updated *Section : Reading the temperature*. Updated *Section : Calculating the actual V_{REF+} voltage using the internal reference voltage*. **VREFBUF:** Added *Section 23.3: VREFBUF trimming*. Updated *Section 23.4.2: VREFBUF calibration control register (VREFBUF_CCR)*. **RNG:** Updated *Section 28.1: Introduction*. Updated *Section 28.3.3: Random number generation*. Updated *Section 28.6.3: Data collection*. **PKA:** Added *Section 32.3.3: PKA reset and clocks*. Updated *Table 248: Arithmetic comparison*. Updated *Table 259: Modular exponentiation computation times*. Updated *Table 260: ECC scalar multiplication computation times*. Updated *Table 261: ECDSA signature average computation times*. Updated *Table 262: ECDSA verification average computation times*. Updated *Table 264: Montgomery parameters average computation times*. **USART:** Updated *Table 44.8.6: USART guard time and prescaler register (USART_GTPR)*. **LPUART:** Added *Table 45.3: LPUART implementation*. **FDCAN:** Updated *Figure 546: Message RAM configuration*.

Table 454. Document revision history (continued)

Date	Revision	Changes
29-Apr-2020	6	**Memory and bus architecture:** Updated *Section 2: Memory and bus architecture*. Updated *Section 2.1: System architecture*. Updated *Section 2.2: TrustZone® security architecture*. Updated *Table 1: Example of memory map security attribution versus SAU regions configuration*. Updated *Section 2.2.1: Default TrustZone security state*. Updated *Section 2.2.2: TrustZone peripheral classification*. Updated *Table 2: Securable peripherals by TZSC*. Updated *Table 3: TrustZone-aware peripherals*. **Memory organization:** Updated *Section 2: Memory and bus architecture*. Updated *Section 2.1: System architecture*. **Embedded SRAM:** Updated *Section 2.4.1: SRAM2 parity check*. Updated *Section 2.4.2: SRAM2 Write protection*. Updated *Section 2.4.4: SRAM2 Erase*. **Embedded Flash memory:** Updated *Section 6.7.2: Readout protection (RDP)* notes. Updated Reset value by ST production value and SWAP_BANK bit description in *Section 6.9.12: Flash option register (FLASH_OPTR)*. Updated Reset value by ST production value in: – *Section 6.9.13: Flash non-secure boot address 0 register (FLASH_NSBOOTADD0R)* – *Section 6.9.14: Flash non-secure boot address 1 register (FLASH_NSBOOTADD1R)* – *Section 6.9.15: Flash secure boot address 0 register (FLASH_SECBOOTADD0R)*. – *Section 6.9.16: Flash bank 1 secure watermak1 register (FLASH_SECWM1R1)*. – *Section 6.9.17: Flash secure watermak1 register 2 (FLASH_SECWM1R2)*. – *Section 6.9.18: Flash WPR1 area A address register (FLASH_WRP1AR)*. – *Section 6.9.19: Flash WPR1 area B address register (FLASH_WRP1BR)*. – *Section 6.9.20: Flash secure watermak2 register (FLASH_SECWM2R1)* – *Section 6.9.21: Flash secure watermak2 register 2 (FLASH_SECWM2R2)* – *Section 6.9.22: Flash WPR2 area A address register (FLASH_WRP2AR)* – *Section 6.9.23: Flash WPR2 area B address register (FLASH_WRP2BR)* – *Table 51: Flash interface - register map and reset values* **ICACHE:** Added *Section 7.3: ICACHE implementation*. Updated *Figure 18: ICACHE block diagram*. Updated *Section 7.4.3: ICACHE TAG memory*. Updated *Table 53: TAG memory dimensioning parameters for n-way set associative operating mode (default)*. Updated *Figure 19: ICACHE TAG and data memories functional view*. Updated *Table 54: TAG memory dimensioning parameters for direct mapped cache mode*. Updated *Section 7.4.5: ICACHE enable*. Updated *Table 59: ICACHE register map and reset values*.

Table 454. Document revision history (continued)

Date	Revision	Changes
29-Apr-2020	6 (continued)	**SYSTEM SECURITY:** Updated *Section 4.6.6: Managing security in TrustZone®-aware peripherals* 'General-purpose I/Os (GPIO)' paragraph. Added *Table 16: Summary of the I/Os that can be secured and connected to a non-secure peripheral*. **PWR:** Updated *Section 8.2.2: Embedded SMPS step down converter*: Updated *Table 60: SMPS modes summary* note 1. Removed below the table. Updated *Section 8.2.4: SMPS step down converter versus low-power mode*. Updated *Table 62: SMPS step down converter versus low-power modes*. Added notes below the table. Moved *Section 8.2.1: Voltage regulator* before *Section 8.2.2: Embedded SMPS step down converter*. Updated *Section 8.2.6: VDD12 domain and external SMPS*. **SYSCFG:** Updated *Table 88: TrustZone security and privilege register accesses*. **TSC**: Updated *Section 27.3.2: Surface charge transfer acquisition overview*. Updated *Section 27.3.4: Charge transfer acquisition sequence*. Updated *Figure 189: Charge transfer acquisition sequence*. **PKA:** Updated *Section 32.7.4: PKA RAM* adding note. **IRTIM:** Added *Figure 391: IRTIM internal hardware connections with TIM16 and TIM17*. **USART:** Replaced 'microcontroller' by 'device' in the whole document. Updated *Section 44.4: USART implementation* and *Section 45.3: LPUART implementation*. Updated PSC bitfield description in USART_GTPR register. Updated SBKF bit description in USART_ISR and LPUART_ISR registers. Updated decimal and hexadecimal notation for values in *Section : How to derive USARTDIV from USART_BRR register values*. **SDMMC:** Updated *Section 48.1: SDMMC main features*. **DEBUG:** Updated *Section 52.11.1: DBGMCU registers*: – DBGMCU identity code register (DBGMCU_IDCODE) – DBGMCU APB1 freeze register 1 (DBGMCU_APB1FZR1) – DBGMCU APB1 freeze register 2 (DGBMCU_APB1FZR2) Updated *Table 453: DBGMCU register map and reset values*.

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Dec-2020	7	**Updated:** **System security:** – *Section 4.4.1: Introduction* – *Section 4.4.3: Immutable root of trust in system Flash memory* – *Section : Securing peripherals with TZSC* – *Table 10: MPCWMx instances* – *Section : General-purpose I/Os (GPIO)* – *Table 16: Summary of the I/Os that can be secured and connected to a non-secure peripheral* – *Section : Extended interrupts and event controller (EXTI)* – *Section 4.6.7: Activating TrustZone® security* – *Figure 8: Flash memory secure hide protection (HDP) area* – *Section : Response to tampers* – *Section 4.9.1: Introduction* – *Section 4.11.3: Recommended option byte settings* – *Figure 13: External Flash memory protection using SFI* **Embedded Flash memory (FLASH):** – *Section 6.7.2: Readout protection (RDP)* – *Section 6.9.12: Flash option register (FLASH_OPTR)* – *Table 51: Flash interface - register map and reset values* **Power control (PWR):** – *Section 8.2.2: Embedded SMPS step down converter* – *Section 8.2.6: VDD12 domain and external SMPS* – *Section 8.3.1: Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR)* – *Section 8.3.2: Programmable voltage detector (PVD)* – *Table 65: Functionalities depending on the working mode* – *Section : I/O states in Low-power run mode* – *Section 8.6.3: Power control register 3 (PWR_CR3)* **Clock recovery system (CRS):** – *Section 10.7.1: CRS control register (CRS_CR)* **Direct memory access controller (DMA):** – *Section 14.6.3: DMA channel x configuration register (DMA_CCRx)* – *Section 14.6.4: DMA channel x number of data to transfer register (DMA_CNDTRx)* – *Section 14.6.5: DMA channel x peripheral address register (DMA_CPARx)* – *Section 14.6.6: DMA channel x memory 0 address register (DMA_CM0ARx)* – *Section 14.6.7: DMA channel x memory 1 address register (DMA_CM1ARx)* **DMA request multiplexer (DMAMUX):** – *Section 15.4.6: DMAMUX request line multiplexer* **Octo-SPI interface (OCTOSPI):** – Major updates in the whole section **Analog-to-digital converters (ADC):** – *Section 21.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)* – *Section 21.4.7: Single-ended and differential input channels* – *Figure 101: Stopping ongoing regular and injected conversions*

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Dec-2020	7 (continued)	**Update (continued):** **Analog-to-digital converters (ADC) (continued):** – *Section 21.4.19: Injected channel management* – *Figure 127: AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)* – *Figure 130: Analog watchdog guarded area* – *Section 21.4.34: Monitoring the internal voltage reference* **Digital-to-analog converter (DAC):** – In this section, DAC_OUTx was replaced by DACxOUTy – *Figure 161: Dual-channel DAC block diagram* – *Table 176: DAC input/output pins* **Voltage reference buffer (VREFBUF):** – *Section 23.4.2: VREFBUF calibration control register (VREFBUF_CCR)* **AES hardware accelerator (AES):** – *Section : GCM processing* – *Section 29.5: AES interrupts* – *Section 29.7.1: AES control register (AES_CR)* – *Section 29.7.17: AES suspend registers (AES_SUSPxR)* – *Table 225: AES register map and reset values* **Hash processor (HASH):** – In this section “NBLW” is now referred to as “NBLW[4:0]” – *Section 30.4.5: Message digest computing* – *Section : HMAC processing* – *Section 30.7.1: HASH control register (HASH_CR)* – *Section 30.7.7: HASH context swap registers* – *Table 230: HASH register map and reset values* **On-the-fly decryption engine (OTFDEC):** – *Section : OTFDEC architecture* – *Section 31.5.1: OTFDEC initialization process* – *Section 31.6.1: OTFDEC control register (OTFDEC_CR)* – *Section 31.6.2: OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR)* – *Section 31.6.3: OTFDEC region x configuration register (OTFDEC_RxCFGR)* **Public key accelerator (PKA):** – *Section 32.3.4: PKA public key acceleration* – *Section : Montgomery space and fast mode operations* – *Section 32.3.6: PKA procedure to perform an operation* – *Section : Using precomputed Montgomery parameters (PKA fast mode)* – *Section 32.5.1: Supported elliptic curves* – *Table 258: Family of supported curves for ECC operations* – *Table 259: Modular exponentiation computation times* – *Table 263: Point on elliptic curve Fp check average computation times* – *Table 264: Montgomery parameters average computation times* – *Table 265: PKA interrupt requests* – *Section 32.7.2: PKA status register (PKA_SR)* – *Section 32.7.3: PKA clear flag register (PKA_CLRFR)*

Table 454. Document revision history (continued)

Date	Revision	Changes
12-Dec-2020	7 (continued)	**Update (continued):** **General-purpose timers (TIM2/TIM3/TIM4/TIM5):** – *Figure 313: Control circuit in external clock mode 2* **Universal synchronous/asynchronous receiver transmitter (USART/UART):** – *Section 44.2: USART main features* – *Table 336: USART / LPUART features* – *Section 44.5.7: USART baud rate generation* – *Figure 451: RS232 RTS flow control* – *Figure 452: RS232 CTS flow control* – *Section 44.6: USART in low-power modes* – *Section 44.7: USART interrupts* – *Section 44.8.8: USART request register (USART_RQR)* – *Section 44.8.9: USART interrupt and status register [alternate] (USART_ISR)* – *Section 44.8.10: USART interrupt and status register [alternate] (USART_ISR)* **Low-power universal asynchronous receiver transmitter (LPUART):** – *Table 345: USART / LPUART features* – *Table 346: Error calculation for programmed baud rates at lpuart_ker_ck_pres = 32,768 KHz* – *Section 45.5: LPUART in low-power modes* – *Table 350: Effect of low-power modes on the LPUART* – *Table 351: LPUART interrupt requests* **Serial peripheral interface (SPI):** – *Table 353: STM32L552xx and STM32L562xx SPI implementation* – *Section 46.4.2: Communications between one master and one slave* **Serial audio interface (SAI):** – *Table 356: STM32L5 Series SAI features* **Secure digital input/output MultiMediaCard interface (SDMMC):** – *Section 48.9.2: SDMMC clock control register (SDMMC_CLKCR)* **FD controller area network (FDCAN):** – *Section 49.3.3: Message RAM* – *Figure 546: Message RAM configuration* **USB Type-C™ / USB Power Delivery interface (UCPD):** – *Section 51.1: Introduction* – *Section 51.2: UCPD main features* – *Section 51.4.6: UCPD Type-C pull-ups (Rp) and pull-downs (Rd)* – *Section 51.7.1: UCPD configuration register 1 (UCPD_CFGR1)* – *Section 51.7.4: UCPD control register (UCPD_CR)* – *Section Table 430.: UCPD register map and reset values* **Added:** – *Table 15: Summary of the I/Os that cannot be connected to a non-secure peripheral when secure* – *Section 51.7.3: UCPD configuration register 3 (UCPD_CFGR3)* **Deleted:** – On *Section 45.3: LPUART implementation*: Table USART /LPUART features – On *Section 48.4: SDMMC functional description*: Section Hardware flow control

Index

D

E

F

G

H

I

L

M

O

P

R

S

T

U

V

W